SIEMENS

Application Example• 08-2015

Simple Examples for the Web Server of SIMATIC S7-1200 / S7-1500

STEP 7 Basic (TIA Portal), STEP 7 Professional (TIA Portal)

https://support.industry.siemens.com/cs/ww/en/68011496

Warranty and Liability

Note

The Application Examples are not binding and do not claim to be complete with regard to configuration, equipment or any contingencies. The Application Examples do not represent customer-specific solutions. They are only intended to provide support for typical applications. You are responsible for the correct operation of the described products. These Application Examples do not relieve you of the responsibility of safely and professionally using, installing, operating and servicing equipment. When using these Application Examples, you recognize that we cannot be made liable for any damage/claims beyond the liability clause described. We reserve the right to make changes to these Application Examples at any time and without prior notice. If there are any deviations between the recommendations provided in this Application Example and other Siemens publications – e.g. Catalogs – the contents of the other documents shall have priority.

We do not accept any liability for the information contained in this document.

Any claims against us – based on whatever legal reason – resulting from the use of the examples, information, programs, engineering and performance data etc., described in this application example shall be excluded. Such an exclusion shall not apply in the case of mandatory liability, e.g. under the German Product Liability Act ("Produkthaftungsgesetz"), in case of intent, gross negligence, or injury of life, body or health, guarantee for the quality of a product, fraudulent concealment of a deficiency or breach of fundamental contractual obligations ("wesentliche Vertragspflichten"). The damages for a breach of a substantial contractual obligation are, however, limited to the foreseeable damage, typical for the type of contract, except in the event of intent or gross negligence or injury to life, body or health. The above provisions do not imply a change of the burden of proof to your detriment.

Any form of duplication or distribution of these application examples or excerpts hereof is prohibited without the expressed consent of Siemens AG.

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit http://www.siemens.com/industrialsecurity.

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit https://support.industry.siemens.com.

Table of Contents

1 Preface

Goal of the application example

This application example shows you different ways of expanding your web pages illustrated by means of simple examples.

Core contents of the application example

This document covers the following core topics:

- Reading and writing with different tag types
- Displaying the time
- Output of arrays (S7-1500 only)
- Using the ENUM data type
- HTTP redirection following a fault (S7-1500 only)
- Language switching on web pages
- Transmitting data without reloading pages
- Recording a PLC tag with a graph

Advantages

Integrated web server
The standard web pages for simple display of services and diagnostic information are activated with a click. Additionally, individually designed, user-defined web pages can be generated.

Location-independent
The web page can be called up world-wide via a standard Internet browser.

Application example
Universal use of the application example for SIMATIC S7-1200 and S7-1500

Benefit

No additional hardware and software required.

Access to the web server is possible across large distances via mobile communication devices such as tablet PC, smart phone, etc.

Note The application example as well as the web server should not and cannot replace an HMI system.

2 Hardware and Software Components Used

The application example was created with the following components:

Hardware components

Note

For this application example you require the latest firmware version of the CPU. Depending on the CPU type, the following entries contain links to the corresponding downloads:

- S7-1500: https://support.industry.siemens.com/cs/ww/en/view/78301349
- S7-1200: https://support.industry.siemens.com/cs/ww/en/view/107539750

Table 2-1

Component	No.	Article number	Note
CPU 1511-1PN/DP	1	6ES7516-3AN00-0AB0	
CPU 1214C DC/DC/DC		214-1AG40-0XB0	
PG/PC with Ethernet interface	1	-	-
IE FC TP STANDARD CABLE	1	6XV1840-2AH10	Connecting cable IE; minimum order quantity 20 m
RJ45 connector	2	6GK1901-1BB10-2AA0	Can be assembled

Software components

Table 2-2

Component	No.	Article number	Note
SIMATIC STEP 7 Professional V13 SP1	1	6ES7822-1..03-..	-
Software tool for creating HTML files, e.g. Frontpage, Notepad++, …	1	-	-
Web browser, e. g. Internet Explorer, Mozilla Firefox[1]	1	-	Application example optimized for Internet Explorer 11.0.

[1] The following web browsers were explicitly tested for communication with the CPU:

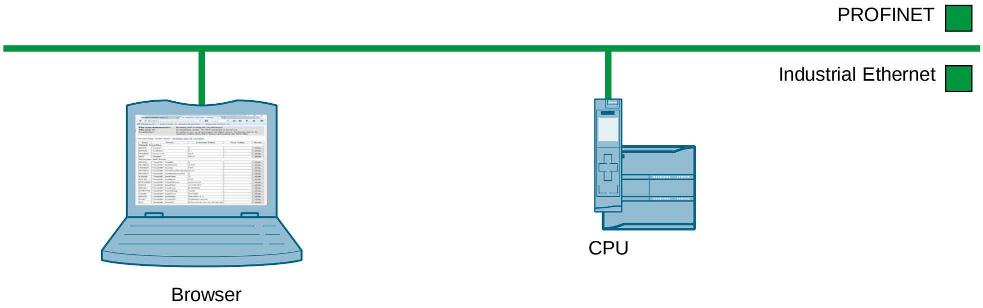
- Internet Explorer (version 11.0)
- Mozilla Firefox (version 31.0)

Note Application example optimized for Internet Explorer 11.0. When using other browsers, adjustments may have to be made.

Overall setup

The single program examples consist of one S7 program and HTML files displayed by means of a browser (web pages).

Figure 2-1 Overall setup



Example files and projects

The following list includes all files and projects that are used in this example.

Table 2-3

Component	Comments
68011496_simple_examples_for_webserver_CODE_v10.zip	The zip file contains the STEP 7 project with the related HTML file. The HTML file with the associated files, are located in the \html directory.
68011496_simple_examples_for_webserver_en pdf	This document.

3 Principles of Standard Web Pages

Requirements

In STEP 7 the following settings are required in the CPU properties:

- The web server has to be activated.
- If you need secure access to the standard web pages, enable the “Permit access only with HTTPS” check box.
- Automatic refreshing of the standard web pages is enabled. The refresh interval is preset to 10 seconds and can lie in the range between 1 and 999.

Access via HTTP or HTTPS

With the URL “http://ww.xx.yy.zz” or “https://ww.xx.yy.zz” you get access to the standard web pages. “ww.xx.yy.zz” corresponds to the IP address of the CPU.

HTTPS is used for the encryption and authentication of the communication between browser and web server. When the “Permit access only with HTTPS” check box is enabled, calling the web pages of the CPU is only possible via HTTPS.

When the browser reports a certificate error, proceed as described in the following FAQ:

https://support.industry.siemens.com/cs/ww/en/view/63314183

Login

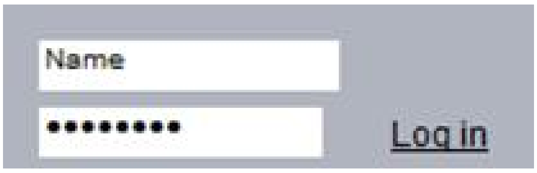
The user “Everybody” is set by default in each SIMATIC S7-1200/1500 controller. This user has limited access rights (no access to user-defined web pages) and does not have a password.

To have full access to the user-defined web pages, you have to log on with a user having the appropriate access rights.

You can parameterize the users, passwords and access rights with STEP 7 in the properties of the S7-1200/1500 controller.

The login input boxes are located in the upper left corner of each standard web page of the S7-1200/1500 controller.

Figure 3-1 Login window



Standard web pages of SIMATIC S7-1200 / S7-1500

The web server of S7-1200 / S7-1500 already offers plenty of information regarding the respective CPU via integrated standard web pages.

A detailed description of the setup of the standard web pages is available in the S7-1500 Web Server Function Manual; it is not subject of this application document.

4 Principles of User-defined Pages

Basic information on user-defined web pages is given in the application example “Creating and using user-defined web pages on S7-1200”: https://support.industry.siemens.com/cs/ww/en/view/58862931

5 Reading and Writing with Different Tag Types

5.1 Automation task

The automation task is to read and write to tags of various data types.

Note The DTL data type is supported by the S7-1500 only.

5.2 Automation solution

Requirements for the automation task

Two HTML pages have to be programmed:

- Programming one HTML page that can be used to read and write to variables of different types.
- Programming one HTML page that can be used to read special tags.

5.3 Functional mechanisms

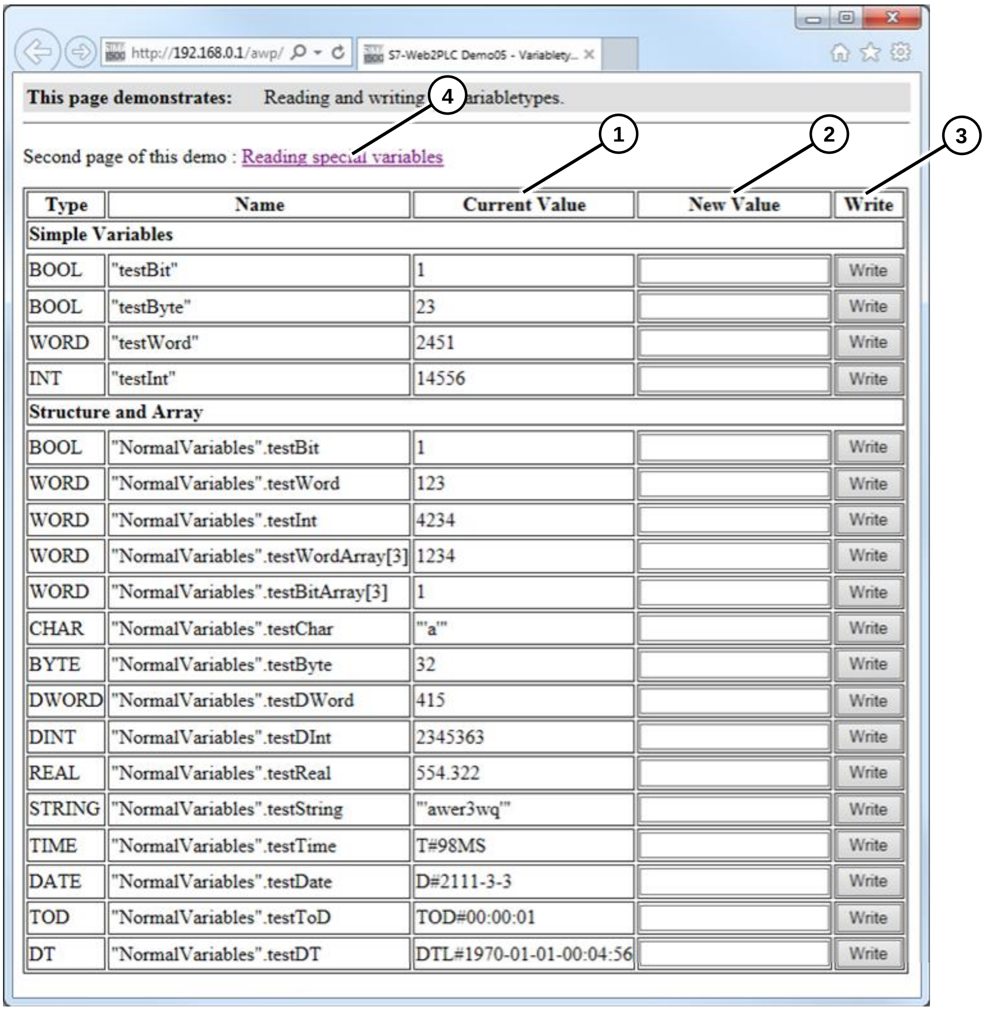
Overall setup

The program of this example consists of one S7 program and HTML files displayed by means of a browser (web pages).

Setup of the web pages

Setup of the web pages for reading and writing PLC tags:

Figure 5-1 web page reading/writing tags



This page demonstrates: Reading and writing variabletypes.

Second page of this demo : Reading special variables

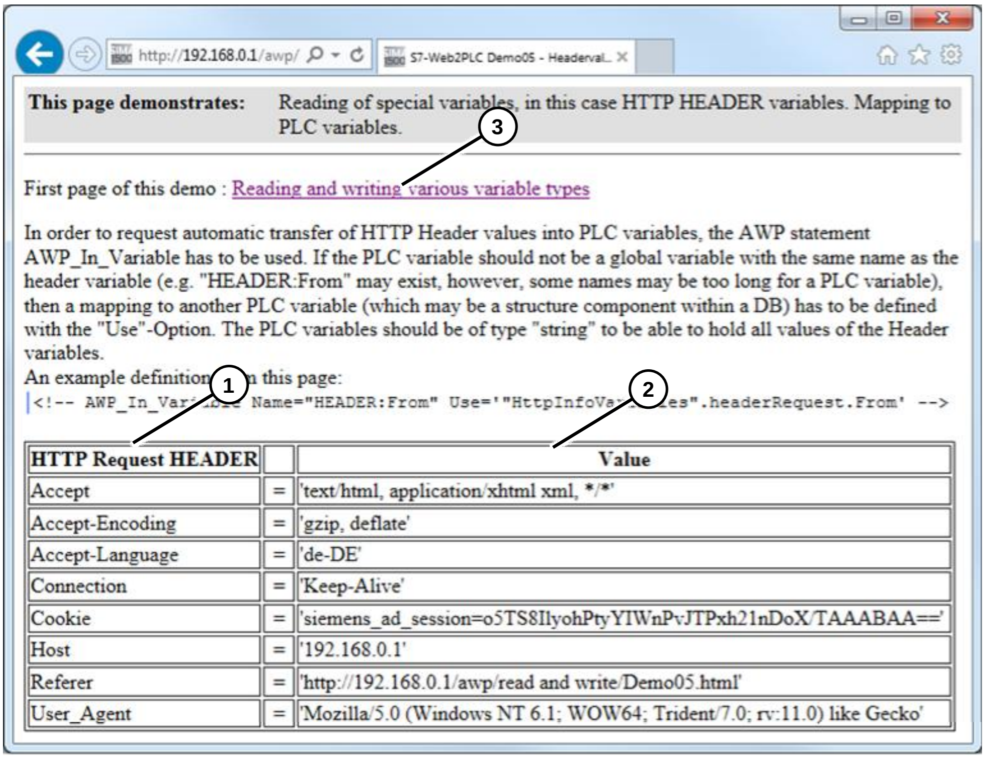
Type	Name	Current Value	New Value	Write
Simple Variables				
BOOL	"testBit"	1		Write
BOOL	"testByte"	23		Write
WORD	"testWord"	2451		Write
INT	"testInt"	14556		Write
Structure and Array				
BOOL	"NormalVariables".testBit	1		Write
WORD	"NormalVariables".testWord	123		Write
WORD	"NormalVariables".testInt	4234		Write
WORD	"NormalVariables".testWordArray[3]	1234		Write
WORD	"NormalVariables".testBitArray[3]	1		Write
CHAR	"NormalVariables".testChar	"'a'"		Write
BYTE	"NormalVariables".testByte	32		Write
DWORD	"NormalVariables".testDWord	415		Write
DINT	"NormalVariables".testDInt	2345363		Write
REAL	"NormalVariables".testReal	554.322		Write
STRING	"NormalVariables".testString	"'awer3wq'"		Write
TIME	"NormalVariables".testTime	T#98MS		Write
DATE	"NormalVariables".testDate	D#2111-3-3		Write
TOD	"NormalVariables".testToD	TOD#00:00:01		Write
DT	"NormalVariables".testDT	DTL#1970-01-01-00:04:56		Write

Table 5-1

Position number	Description
1	This column shows the current values of the tag.
2	The value to be written has to be entered in this column.
3	This column contains the buttons for transmitting the values to the CPU.
4	This link takes the user to the web page with the special tags.

Setup of the web page for reading special tags:

Figure 5-2 Reading/writing tags web page



This page demonstrates: Reading of special variables, in this case HTTP HEADER variables. Mapping to PLC variables.

First page of this demo : Reading and writing various variable types

In order to request automatic transfer of HTTP Header values into PLC variables, the AWP statement AWP_In_Variable has to be used. If the PLC variable should not be a global variable with the same name as the header variable (e.g. "HEADER:From" may exist, however, some names may be too long for a PLC variable), then a mapping to another PLC variable (which may be a structure component within a DB) has to be defined with the "Use"-Option. The PLC variables should be of type "string" to be able to hold all values of the Header variables.

An example definition on this page:

```
<!-- AWP_In_Variable Name="HEADER:From" Use='"HttpInfoVariables".headerRequest.From' -->
```

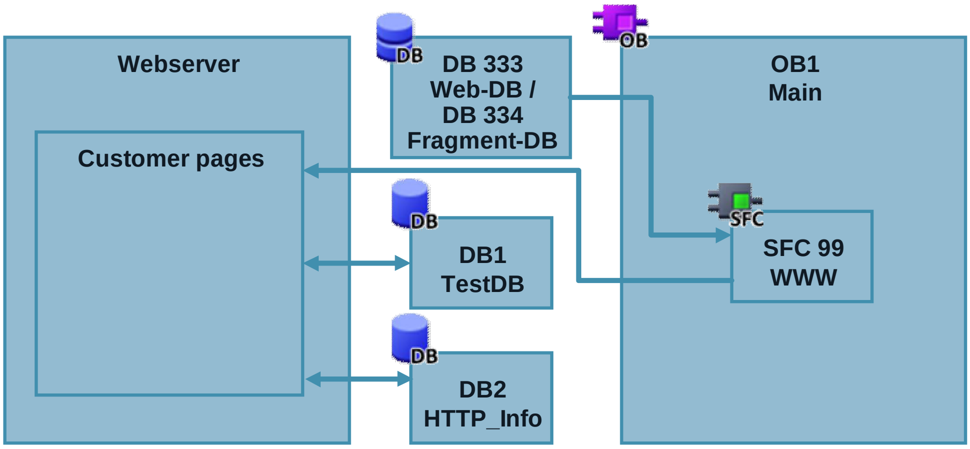
HTTP Request HEADER		Value
Accept	=	'text/html, application/xhtml xml, */*'
Accept-Encoding	=	'gzip, deflate'
Accept-Language	=	'de-DE'
Connection	=	'Keep-Alive'
Cookie	=	'siemens_ad_session=o5TS8IlyohPtyYIWnPvJTPxh21nDoX/TAAABAA=='
Host	=	'192.168.0.1'
Referer	=	'http://192.168.0.1/awp/read and write/Demo05.html'
User_Agent	=	'Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko'

Table 5-2

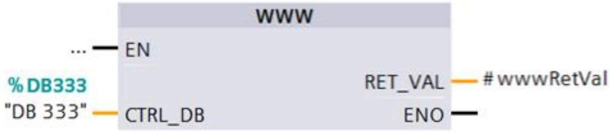
Position number	Description
1	This column contains the names of the special tags.
2	This column contains the information of the special tags.
3	This link takes the user to the web pages for reading and writing PLC tags.

5.3.1 Functional principle of the S7 program

Figure 5-3 S7 program reading/writing tags



Functioning of OB1

No.	Function
1	 WWW ... EN %DB333 "DB 333" CTRL_DB RET_VAL #wwwRetVal ENO The WWW function (SFC 99) is called here. The web DB (DB 333) is connected to this function. The setup of the user pages is stored in the web DB and in the fragment DB(s). Based on this information, the WWW function (SFC 99) creates the user pages.

Content of DB1

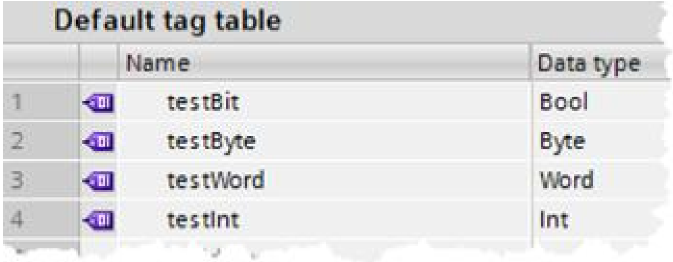
Tags are defined in “TestDB” (DB1) that can be read and written using the web page with the exception of the tag “WWW_RET_VAL”. This tag contains the return value of the WWW function.

Content of DB2

Special tags are stored in DB2 “HTTP_Info”, HEADER_Request tags in the present case. The user can only read these tags through the web page. These tags are written to using the web page via the web server.

5.3.2 Functional principle of the HTML file

<table>
<tr><th>No.</th><th>HTML call</th></tr>
<tr><td>1</td><td>



Default tag table

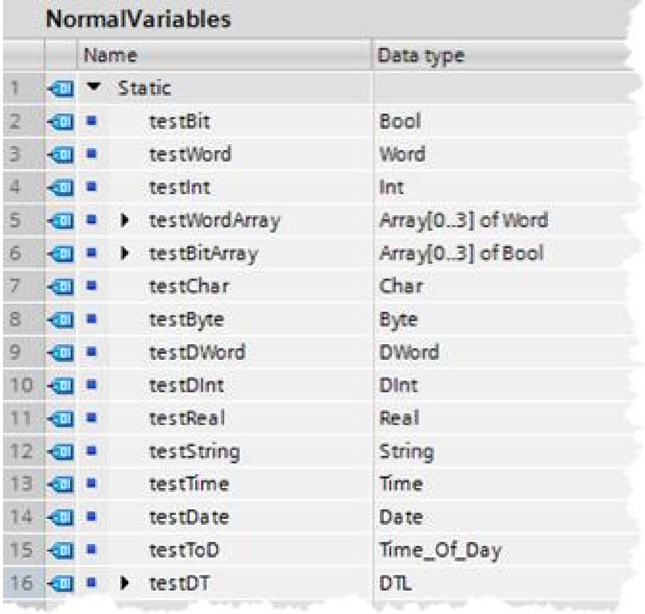
		Name	Data type
1		testBit	Bool
2		testByte	Byte
3		testWord	Word
4		testInt	Int

```html
<!-- AWP_In_Variable Name='"testBit"' -->

<tr>
<form method="post" action="" >
<td>BOOL</td><td>"testBit"</td><td>:="testBit":</td>
<td><input type="text" name='"testBit"' maxlength="8" ></td>
<td><input type="submit" value="Write"></td>
</form>
</tr>
```

To read and write to a PLC tag from the PLC tags (tag table) of a web page, only the tag name is relevant.

</td></tr>
</table>

<table>
<tr><th>No.</th><th>HTML call</th></tr>
<tr><td>2</td><td>



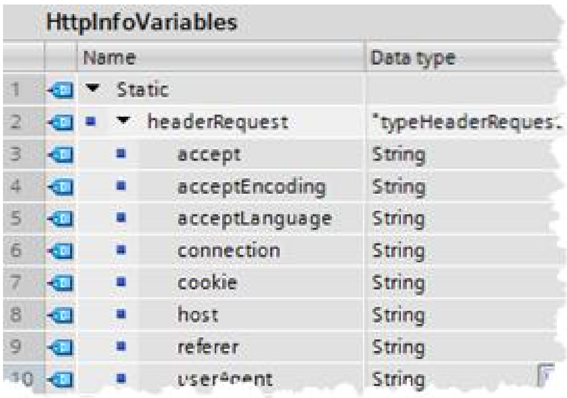
NormalVariables

	Name	Data type
1	▼ Static	
2	testBit	Bool
3	testWord	Word
4	testInt	Int
5	▶ testWordArray	Array[0..3] of Word
6	▶ testBitArray	Array[0..3] of Bool
7	testChar	Char
8	testByte	Byte
9	testDWord	DWord
10	testDInt	DInt
11	testReal	Real
12	testString	String
13	testTime	Time
14	testDate	Date
15	testToD	Time_Of_Day
16	▶ testDT	DTL

```
<!-- AWP_In_Variable Name='"NormalVariables".testBit' -->

<tr>
<form method="post" action="">
<td>BOOL</td>
<td>"NormalVariables".testBit</td>
<td>:="NormalVariables".testBit:</td>
<td>
    <input type="text" name='"NormalVariables".testBit' maxlength="8" >
</td>
<td>
    <input type="submit" value="Write">
</td>
</form>
</tr>
```

To read and write to a tag from and to a DB, the DB name and the tag name are relevant.

</td></tr>
</table>

<table>
<tr><th>No.</th><th>HTML call</th></tr>
<tr><td><b>3</b></td><td>


HttpInfoVariables

<table>
<tr><th></th><th>Name</th><th>Data type</th></tr>
<tr><td>1</td><td>▼ Static</td><td></td></tr>
<tr><td>2</td><td>▼ headerRequest</td><td>"typeHeaderReques</td></tr>
<tr><td>3</td><td>accept</td><td>String</td></tr>
<tr><td>4</td><td>acceptEncoding</td><td>String</td></tr>
<tr><td>5</td><td>acceptLanguage</td><td>String</td></tr>
<tr><td>6</td><td>connection</td><td>String</td></tr>
<tr><td>7</td><td>cookie</td><td>String</td></tr>
<tr><td>8</td><td>host</td><td>String</td></tr>
<tr><td>9</td><td>referer</td><td>String</td></tr>
<tr><td>10</td><td>userAgent</td><td>String</td></tr>
</table>

```
<!-- AWP_In_Variable Name="HEADER:Accept"
    Use='"HttpInfoVariables".headerRequest.accept' -->

:="HttpInfoVariables".headerRequest.accept:
```

The special tags (here: HTTP Request tags) are saved in the DB tags with AWP commands. This DB tag is read by the HTML file.
</td></tr>
</table>

5.4 Operation

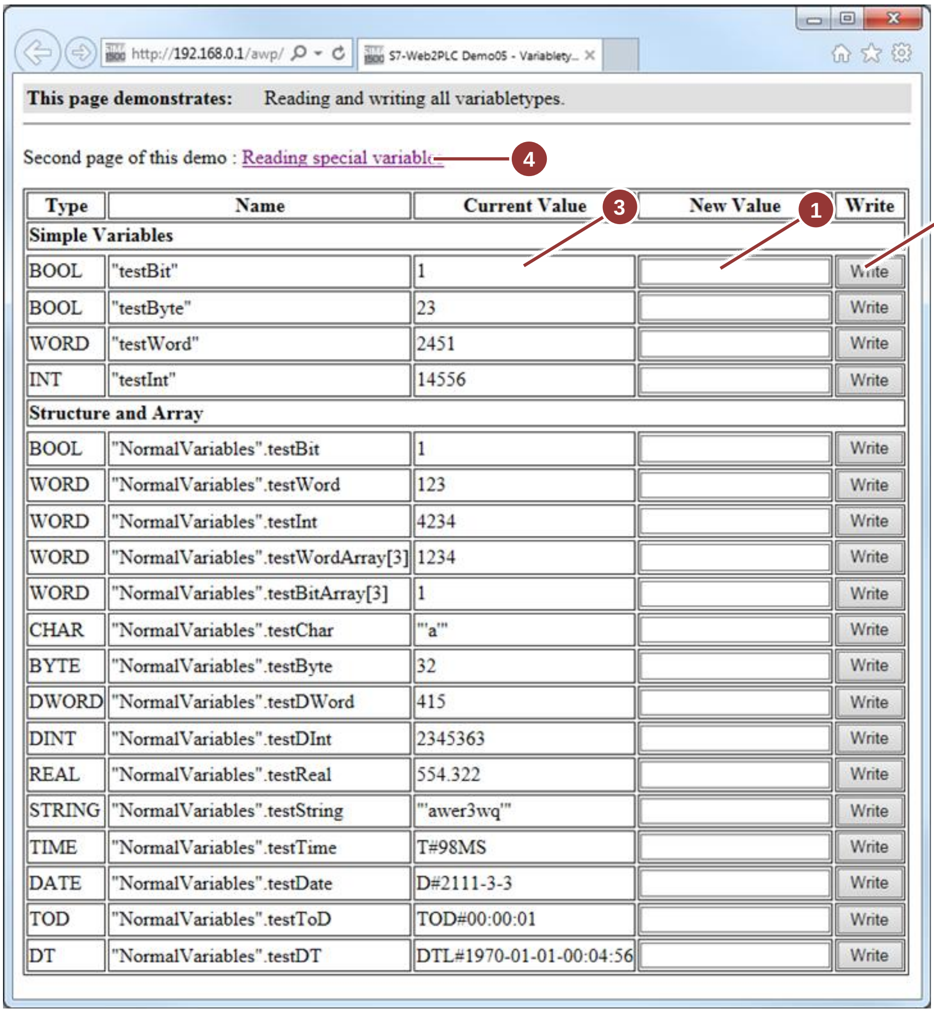
Note The STEP7 program has the number of this chapter.

To get to the user pages, the “admin” user has to be logged in using the password “s7”. See 3. Principles of Standard Web Pages

1. Enter a value corresponding to the data type in the “New Value” column.
2. Click the “Write” button.
3. The value appears in the “Current Value” column.
4. Click the “Reading special variables” link.
5. Read HTTP Request tags are displayed.

The following illustration shows the web page with tags that can be read and written to.

Figure 5-4 Reading/writing tags1 web page



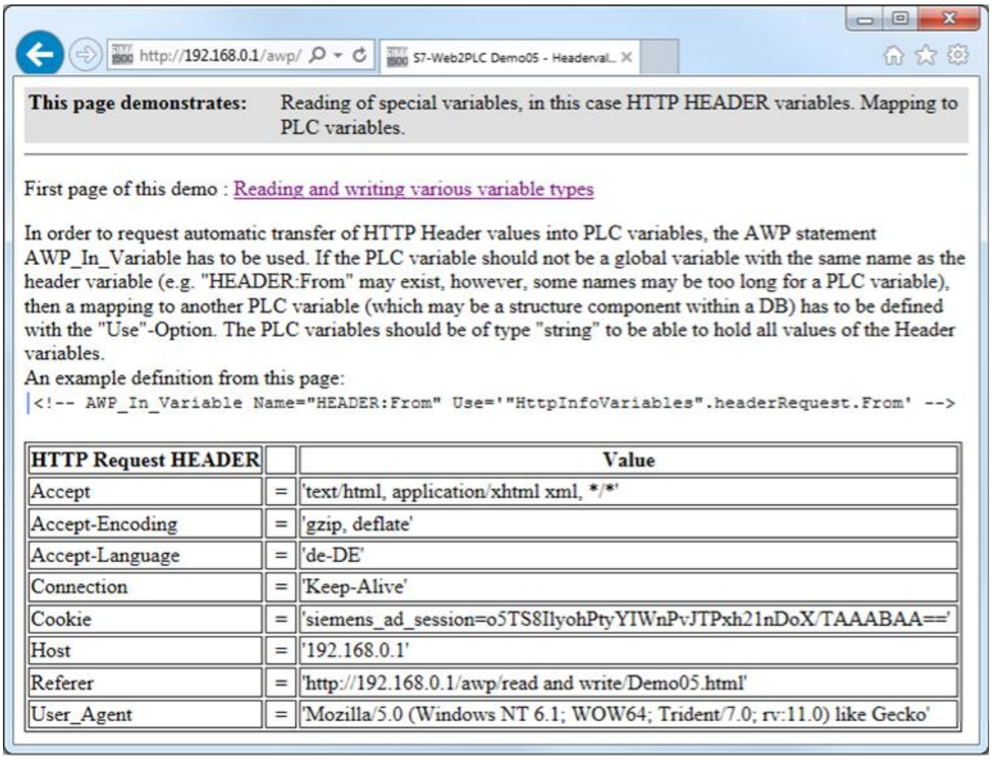
This page demonstrates: Reading and writing all variabletypes.

Second page of this demo : Reading special variable

Type	Name	Current Value	New Value	Write
Simple Variables				
BOOL	"testBit"	1		Write
BOOL	"testByte"	23		Write
WORD	"testWord"	2451		Write
INT	"testInt"	14556		Write
Structure and Array				
BOOL	"NormalVariables".testBit	1		Write
WORD	"NormalVariables".testWord	123		Write
WORD	"NormalVariables".testInt	4234		Write
WORD	"NormalVariables".testWordArray[3]	1234		Write
WORD	"NormalVariables".testBitArray[3]	1		Write
CHAR	"NormalVariables".testChar	'"a"'		Write
BYTE	"NormalVariables".testByte	32		Write
DWORD	"NormalVariables".testDWord	415		Write
DINT	"NormalVariables".testDInt	2345363		Write
REAL	"NormalVariables".testReal	554.322		Write
STRING	"NormalVariables".testString	'"awer3wq"'		Write
TIME	"NormalVariables".testTime	T#98MS		Write
DATE	"NormalVariables".testDate	D#2111-3-3		Write
TOD	"NormalVariables".testToD	TOD#00:00:01		Write
DT	"NormalVariables".testDT	DTL#1970-01-01-00:04:56		Write

The following illustration shows the web page with the read special tags.

Figure 5-5 Reading/writing tags2 web page



This page demonstrates: Reading of special variables, in this case HTTP HEADER variables. Mapping to PLC variables.

First page of this demo : Reading and writing various variable types

In order to request automatic transfer of HTTP Header values into PLC variables, the AWP statement AWP_In_Variable has to be used. If the PLC variable should not be a global variable with the same name as the header variable (e.g. "HEADER:From" may exist, however, some names may be too long for a PLC variable), then a mapping to another PLC variable (which may be a structure component within a DB) has to be defined with the "Use"-Option. The PLC variables should be of type "string" to be able to hold all values of the Header variables.
An example definition from this page:

```
<!-- AWP_In_Variable Name="HEADER:From" Use='"HttpInfoVariables".headerRequest.From' -->
```

HTTP Request HEADER		Value
Accept	=	'text/html, application/xhtml xml, */*'
Accept-Encoding	=	'gzip, deflate'
Accept-Language	=	'de-DE'
Connection	=	'Keep-Alive'
Cookie	=	'siemens_ad_session=o5TS8IlyohPtyYIWnPvJTPxh21nDoX/TAAABAA=='
Host	=	'192.168.0.1'
Referer	=	'http://192.168.0.1/awp/read and write/Demo05.html'
User_Agent	=	'Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko'

6 Displaying the Date and Time

6.1 Automation task

The automation task is to display the time of the CPU on a web page.

6.2 Automation solution

Requirements for the automation task

- Acquiring the time in the STEP 7 program using the function “RD_LOC_T”
- Programming a web page on which the time is displayed
- Updating the time using an HTML file integrated in the HTML file (Iframe).

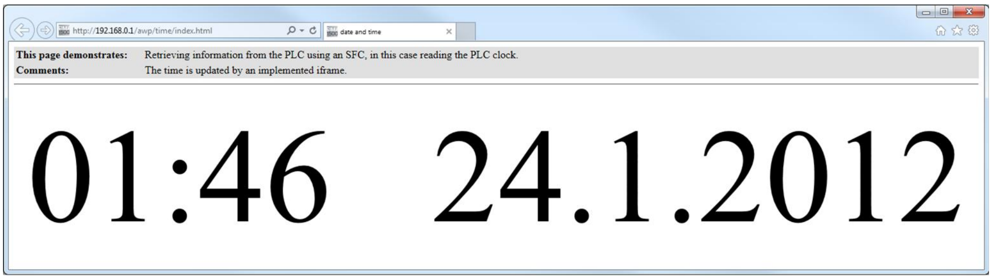
6.3 Functional mechanisms

Overall setup

The program of this example consists of one S7 program and HTML files displayed by means of a browser (web pages).

Setup of the web page

Figure 6-1 time on web page



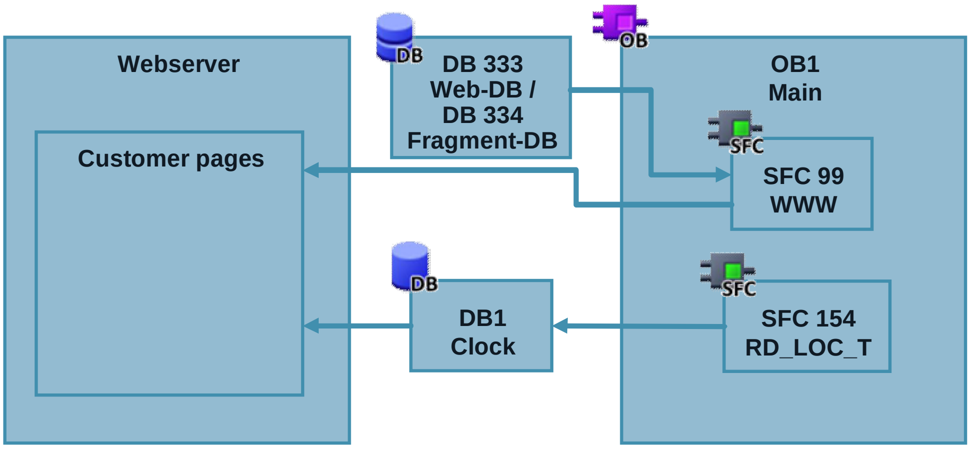
This page demonstrates:	Retrieving information from the PLC using an SFC, in this case reading the PLC clock.
Comments:	The time is updated by an implemented iframe.

01:46 24.1.2012

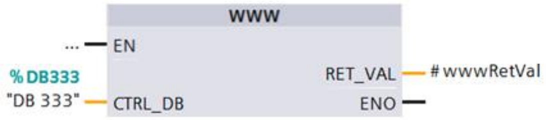
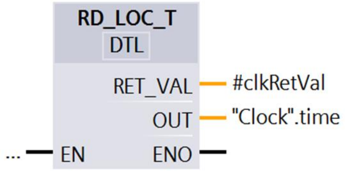
The date and time are displayed in the center of the web page.

6.3.1 Functional principle of the S7 program

Figure 6-2 S7 program, show date and time



Functioning of OB1

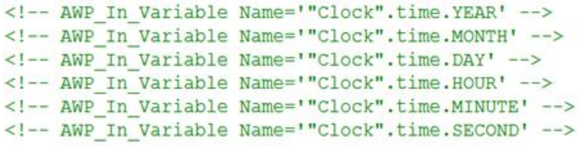
No.	Function
1	 WWW ... — EN %DB333 "DB 333" — CTRL_DB RET_VAL — #wwwRetVal ENO — The WWW function (SFC 99) is called here. The web DB (DB 333) is connected to this function. The setup of the user pages is stored in the web DB and in the fragment DB(s). Based on this information, the WWW function (SFC 99) creates the user pages.
2	 RD_LOC_T DTL RET_VAL — #clkRetVal OUT — "Clock".time ... — EN ENO — Calling the function “RD_LOC_T” (SFC 154). This function reads the CPU time and saves it in the DB “Clock” (DB 1) in the “time” tag.

Content of the clock DB (DB 1)

The time is stored in the format “DTL” in this DB. The web page reads the time from this DB.

6.3.2 Functional principle of the HTML file

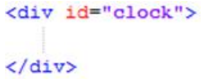
<table>
<tr><th>No.</th><th></th></tr>
<tr><td>1</td><td>



```
<!-- AWP_In_Variable Name='"Clock".time.YEAR' -->
<!-- AWP_In_Variable Name='"Clock".time.MONTH' -->
<!-- AWP_In_Variable Name='"Clock".time.DAY' -->
<!-- AWP_In_Variable Name='"Clock".time.HOUR' -->
<!-- AWP_In_Variable Name='"Clock".time.MINUTE' -->
<!-- AWP_In_Variable Name='"Clock".time.SECOND' -->
```

The AWP commands initialize the tags.

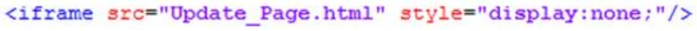
</td></tr>
<tr><td>2</td><td>



```
<div id="clock">

</div>
```

The Iframe (“Update_Page.html” file) writes the time into this div box.

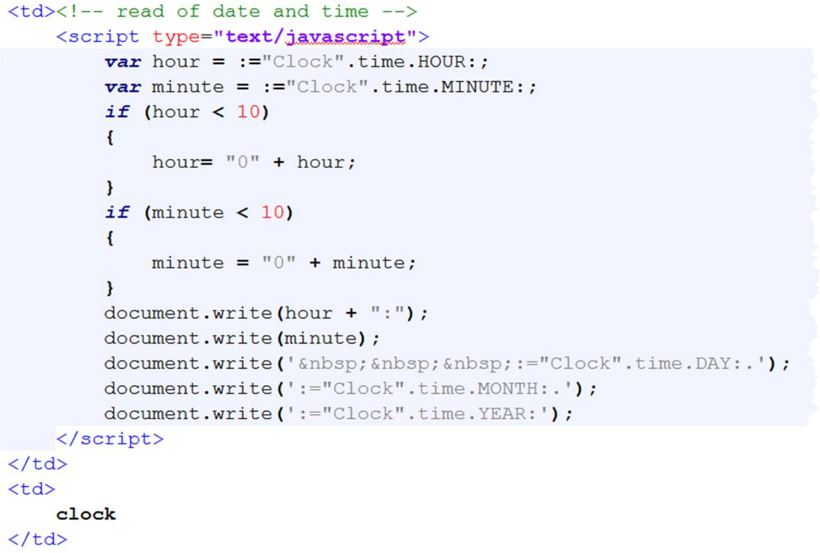
</td></tr>
<tr><td>3</td><td>



```
<iframe src="Update_Page.html" style="display:none;"/>
```

The Iframe has to be integrated into the “index.html” file to be integrated in the function sequence.

</td></tr>
<tr><td>4</td><td>



```
<td><!-- read of date and time -->
    <script type="text/javascript">
        var hour = :="Clock".time.HOUR:;
        var minute = :="Clock".time.MINUTE:;
        if (hour < 10)
        {
            hour= "0" + hour;
        }
        if (minute < 10)
        {
            minute = "0" + minute;
        }
        document.write(hour + ":");
        document.write(minute);
        document.write('&nbsp;&nbsp;&nbsp;:="Clock".time.DAY:.');
        document.write(':="Clock".time.MONTH:.');
        document.write(':="Clock".time.YEAR:');
    </script>
</td>
<td>
    clock
</td>
```

In the file “Update_Page.html” you have to specify which tags to update cyclically and where to store them. (This is implemented through the ID.) In order for the zero before the 3 to be preserved in the time 12:03, an If query is put in front. More detailed information on this updating method is available in the following FAQ:
http://support.automation.siemens.com/WW/view/en/97044123

</td></tr>
</table>

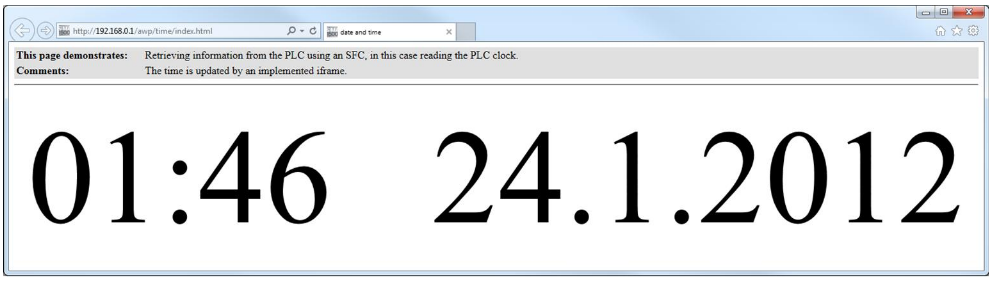
6.4 Operation

Note

> The STEP7 program has the number of this chapter.
>
> To get to the user pages, the “admin” user has to be logged in using the password “s7”. See 3. Principles of Standard Web Pages
>
> To display the correct time on the web page, you have to set the local time in the CPU (Device configuration -> Properties -> General -> Time of day -> Local time -> Time zone) and subsequently set the CPU time (Online & diagnostics -> Functions -> Set time).

The local time of the CPU is displayed on the web page.

Figure 6-3 Web page time



This page demonstrates: Retrieving information from the PLC using an SFC, in this case reading the PLC clock.
Comments: The time is updated by an implemented iframe.

01:46 24.1.2012

7 Reading Arrays (S7-1500 only)

7.1 Automation task

Reading all fields of an array and output in a table

7.2 Automation solution

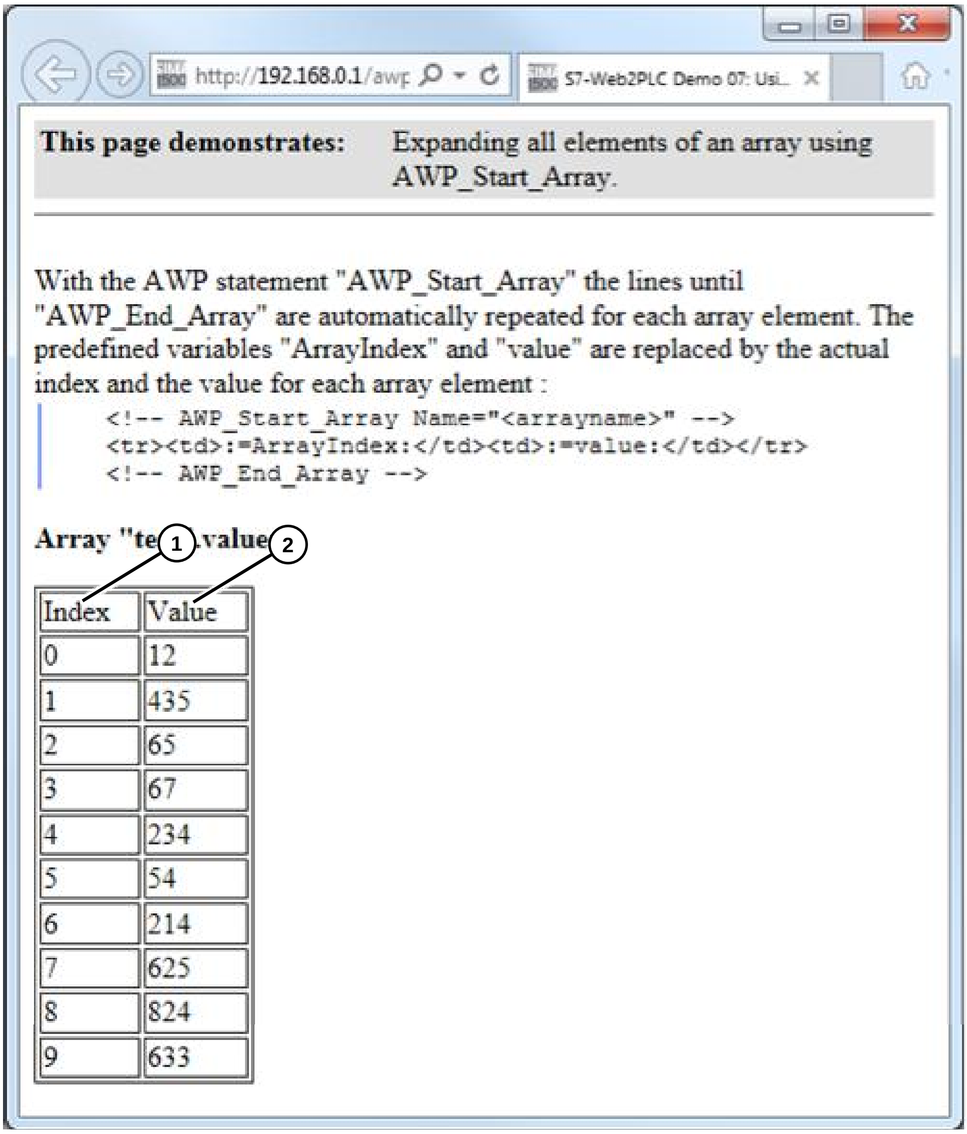
Requirements for the automation task

- Creating an array in a DB of the STEP 7 program
- Using a web page to display all fields of this array structured in a table

7.3 Functional mechanisms

Setup of the web page

Figure 7-1 web page array



This page demonstrates: Expanding all elements of an array using AWP_Start_Array.

With the AWP statement "AWP_Start_Array" the lines until "AWP_End_Array" are automatically repeated for each array element. The predefined variables "ArrayIndex" and "value" are replaced by the actual index and the value for each array element :

```
<!-- AWP_Start_Array Name="<arrayname>" -->
<tr><td>:=ArrayIndex:</td><td>:=value:</td></tr>
<!-- AWP_End_Array -->
```

Array "te[illegible].value

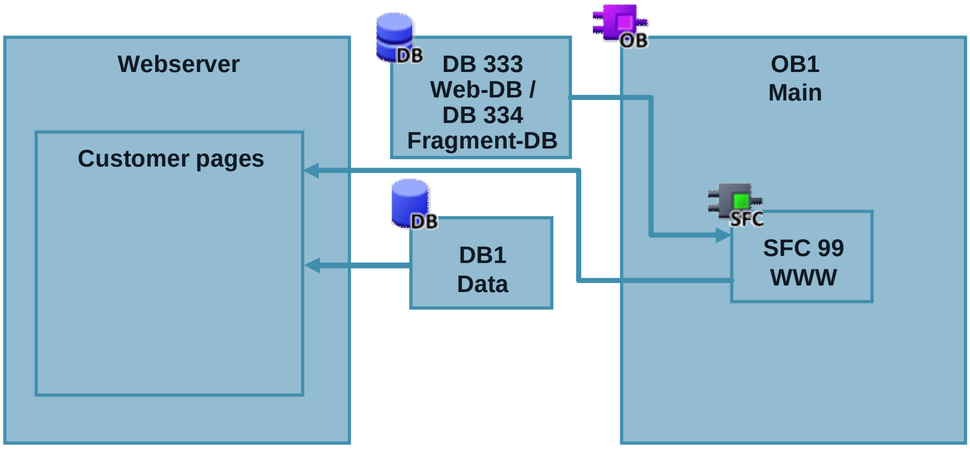
Index	Value
0	12
1	435
2	65
3	67
4	234
5	54
6	214
7	625
8	824
9	633

Table 7-1

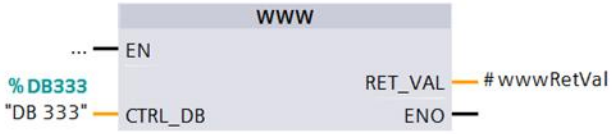
Position number	Description
1	This column shows the array index.
2	This column shows the output value of the array field.

7.3.1 Functional principle of the S7 program

Figure 7-2 Read S7 program array



Functioning of OB1

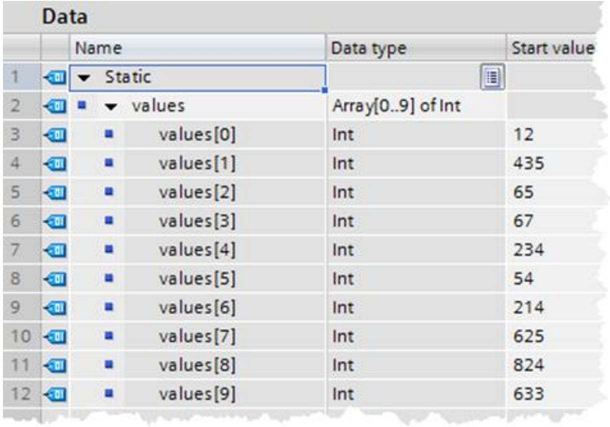
No.	Function
1	 The WWW function (SFC 99) is called here. The web DB (DB 333) is connected to this function. The setup of the user pages is stored in the web DB and in the fragment DB(s). Based on this information, the WWW function (SFC 99) creates the user pages.

Content of the data DB (DB 1)

The array read by the web page is defined in the data DB.

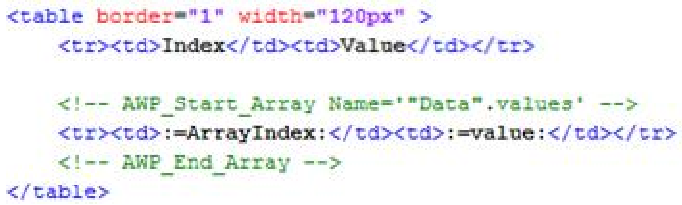
7.3.2 Functional principle of the HTML file

<table>
<tr><th>No.</th><th>Function</th></tr>
<tr><td>1</td><td>



Data

	Name	Data type	Start value
1	▾ Static		
2	▾ values	Array[0..9] of Int	
3	values[0]	Int	12
4	values[1]	Int	435
5	values[2]	Int	65
6	values[3]	Int	67
7	values[4]	Int	234
8	values[5]	Int	54
9	values[6]	Int	214
10	values[7]	Int	625
11	values[8]	Int	824
12	values[9]	Int	633



```
<table border="1" width="120px" >
    <tr><td>Index</td><td>Value</td></tr>

    <!-- AWP_Start_Array Name='"Data".values' -->
    <tr><td>:=ArrayIndex:</td><td>:=value:</td></tr>
    <!-- AWP_End_Array -->
</table>
```

The code example shows how to display array values in a table on the web page.
The AWP command has to be adapted to the respective array name.

</td></tr>
</table>

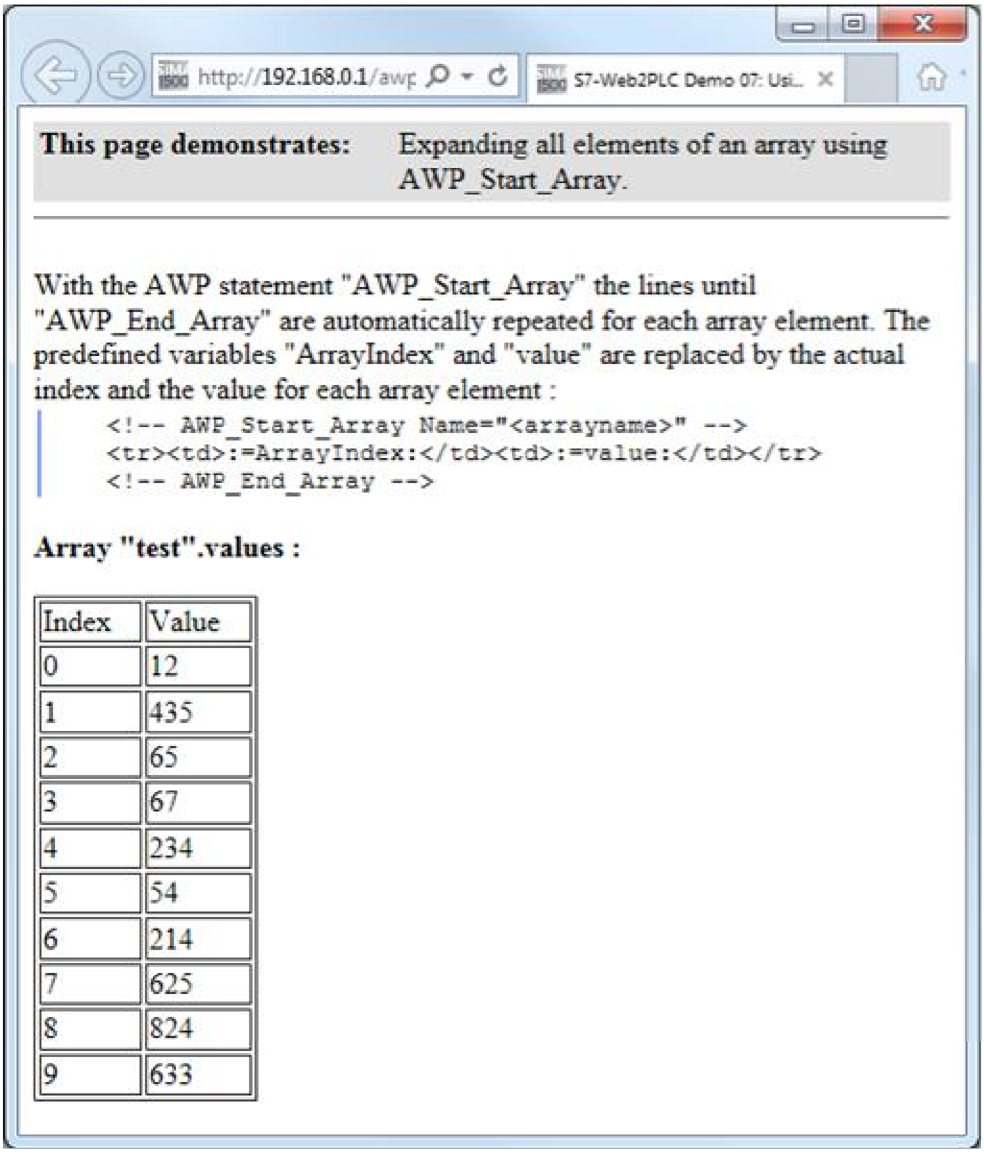
7.4 Operation

Note

> The STEP7 program has the number of this chapter.
>
> To get to the user pages, the “admin” user has to be logged in using the password “s7”. See 3 Principles of Standard Web Pages
>
> The output array values are not updated cyclically.

The web page shows the array values structured in a table.

Figure 7-3 web page array



This page demonstrates: Expanding all elements of an array using AWP_Start_Array.

With the AWP statement "AWP_Start_Array" the lines until "AWP_End_Array" are automatically repeated for each array element. The predefined variables "ArrayIndex" and "value" are replaced by the actual index and the value for each array element :

```
<!-- AWP_Start_Array Name="<arrayname>" -->
<tr><td>:=ArrayIndex:</td><td>:=value:</td></tr>
<!-- AWP_End_Array -->
```

Array "test".values :

Index	Value
0	12
1	435
2	65
3	67
4	234
5	54
6	214
7	625
8	824
9	633

8 Replacing Values of a Tag with Text (ENUM)

8.1 Automation task

The integer value of a tag is to be modified via different buttons on a web page. The values written to the tag are to be linked with different texts. The corresponding texts are to be output.

8.2 Automation solution

Requirements for the automation task

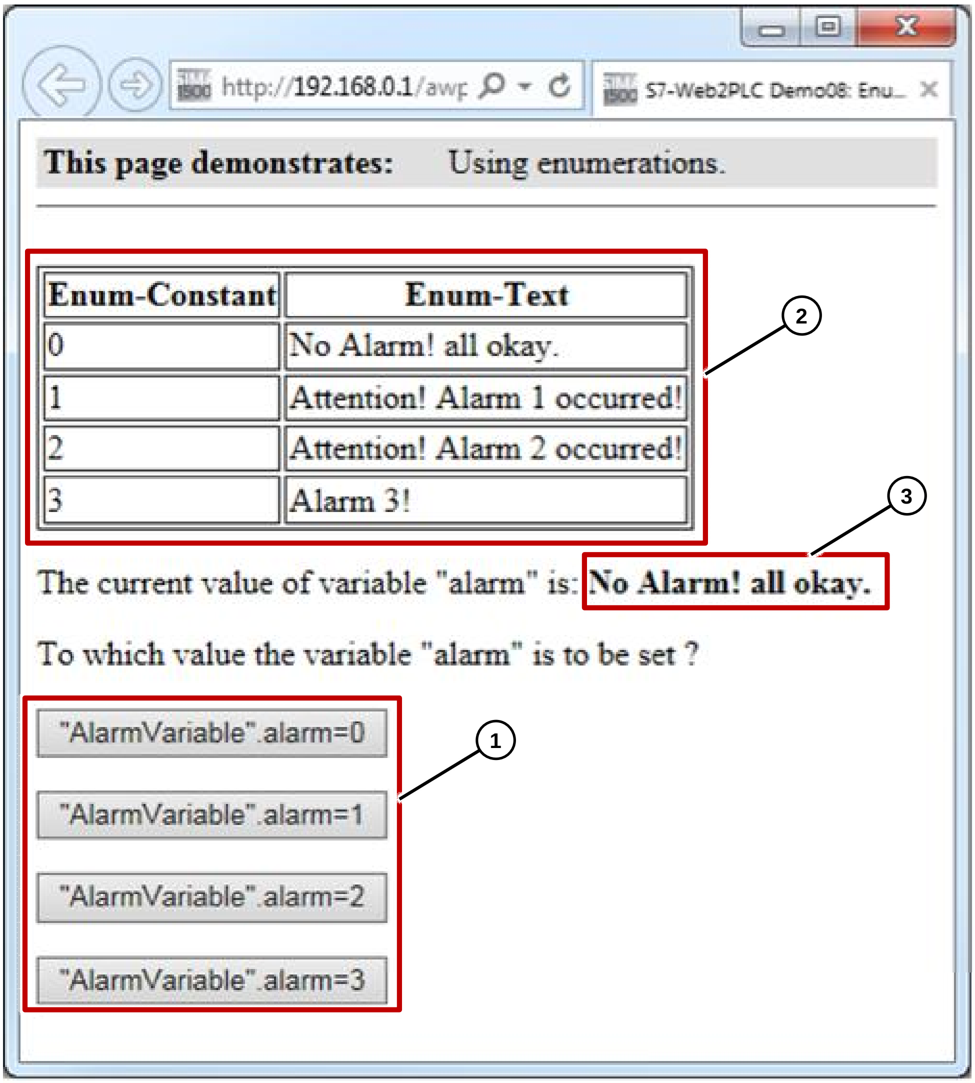
- Writing to and reading a tag defined as ENUM with a web page.
- Defining an ENUM tag.
- Creating a tag in a PLC tag table

Note ENUM tags are defined by the HTML files. The STEP7 program uses a numerical tag. In ENUM tags the numerical values are replaced by a character string.

8.3 Functional mechanisms

Setup of the web page

Figure 8-1 ENUM web page



This page demonstrates: Using enumerations.

Enum-Constant	Enum-Text
0	No Alarm! all okay.
1	Attention! Alarm 1 occurred!
2	Attention! Alarm 2 occurred!
3	Alarm 3!

The current value of variable "alarm" is: **No Alarm! all okay.**

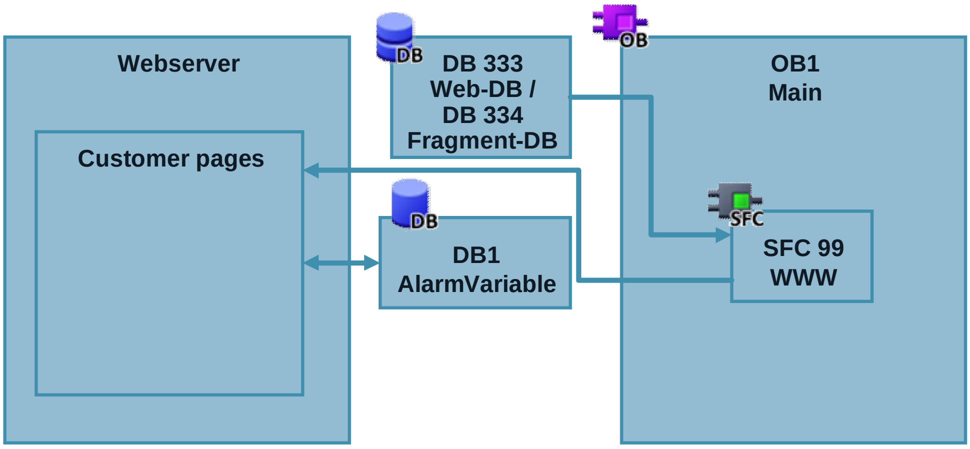
To which value the variable "alarm" is to be set ?

Table 8-1

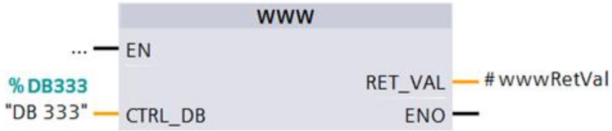
Position number	Description
1	The highlighted buttons can be used to set the “alarm” tag to a value between 0 and 3.
2	This table shows which text is assigned to which numerical value.
3	The corresponding text is output here.

8.3.1 Functional principle of the S7 program

Figure 8-2 S7-ENUM program



Functioning of OB1

No.	Function
1	 WWW ... EN %DB333 "DB 333" CTRL_DB RET_VAL #wwwRetVal ENO The WWW function (SFC 99) is called here. The web DB (DB 333) is connected to this function. The setup of the user pages is stored in the web DB and in the fragment DB(s). Based on this information, the WWW function (SFC 99) creates the user pages.

Content of the Alarm tag DB (DB 1)

The “alarm” tag is stored in this DB and can be read and written to through the web page.

8.3.2 Functional principle of the HTML file

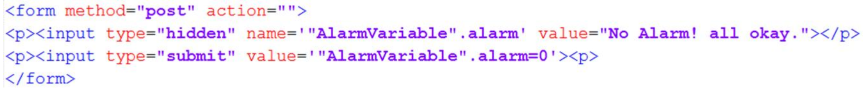
No.	Function

1	`<!-- AWP_Enum_Def Name="MyAlarmEnum" Values='`
	`        0:"No Alarm! all okay.",`
	`        1:"Attention! Alarm 1 occurred!",`
	`        2:"Attention! Alarm 2 occurred!",`
	`        3:"Alarm 3!"' -->`
	`<!-- AWP_Enum_Def Name="abc" Values='0:"null", 1:"one"' -->`
	Defining the ENUM tag in the “_enumdefs.htm” file

2	`<!-- AWP_In_Variable Name='"AlarmVariable".alarm' Enum="MyAlarmEnum" -->` Initializing the ENUM tag

3	`<p>The current value of variable "alarm" is:`
	`<strong> :="AlarmVariable".alarm:</strong></p>`
	The value of the PLC tag is read and displayed as text.

No.	Function

4	 `<form method="post" action="">`
	`<p><input type="hidden" name='"AlarmVariable".alarm' value="No Alarm! all okay."></p>`
	`<p><input type="submit" value='"AlarmVariable".alarm=0'><p>`
	`</form>`
	The assigned text can be used when writing to the PLC tag. The associated number is written to the PLC tag.

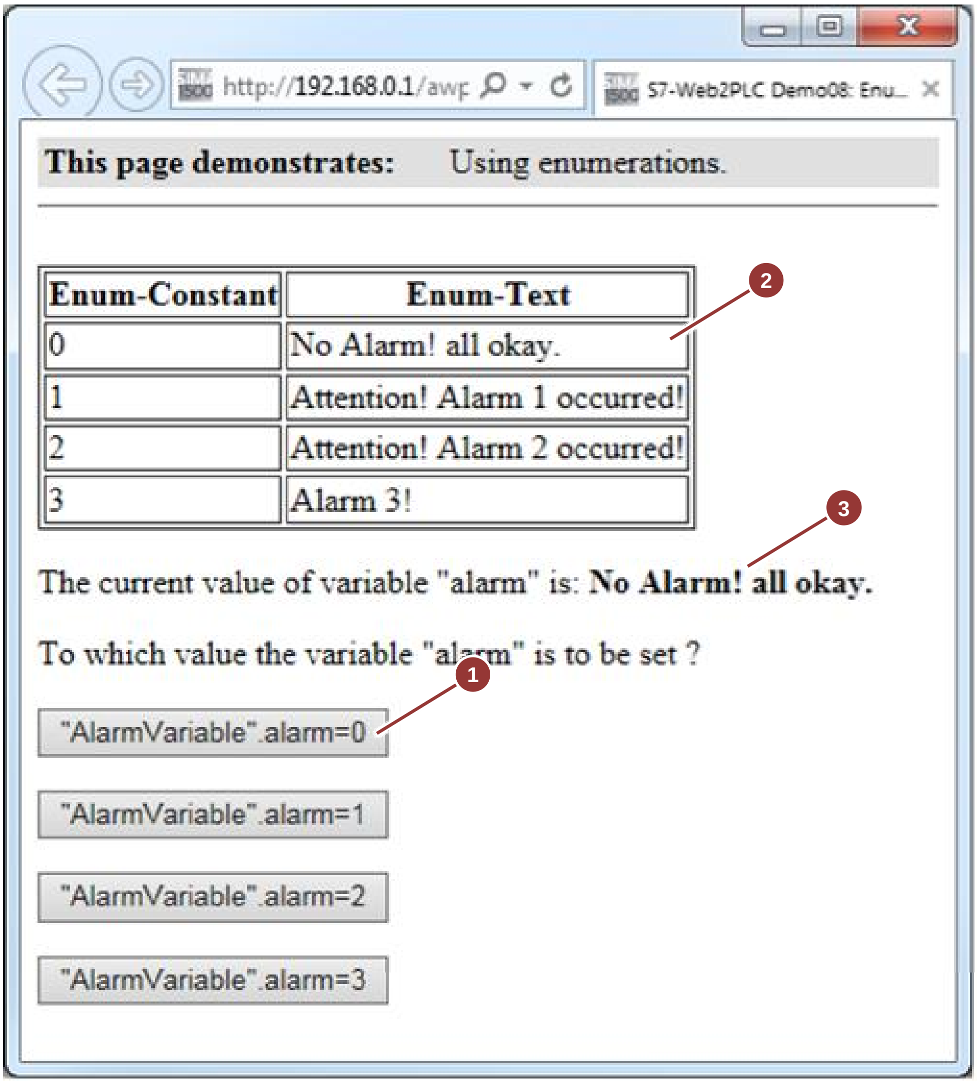
8.4 Operation

Note The STEP7 program has the number of this chapter.

To get to the user pages, the “admin” user has to be logged in using the password “s7”. See 3 Principles of Standard Web Pages

1. Confirm one of the four possible buttons.
2. The table shows which text belongs to which number and hence to which button.
3. The text is output.

Figure 8-3 ENUM web page



This page demonstrates: Using enumerations.

Enum-Constant	Enum-Text
0	No Alarm! all okay.
1	Attention! Alarm 1 occurred!
2	Attention! Alarm 2 occurred!
3	Alarm 3!

The current value of variable "alarm" is: **No Alarm! all okay.**

To which value the variable "alarm" is to be set ?

9 HTTP Redirection Following a Fault (S71500 only)

9.1 Automation task

If a tag has exceeded a specific value, another web page should open displaying a corresponding message. This tag should be settable via an input box on the web page.

9.2 Automation solution

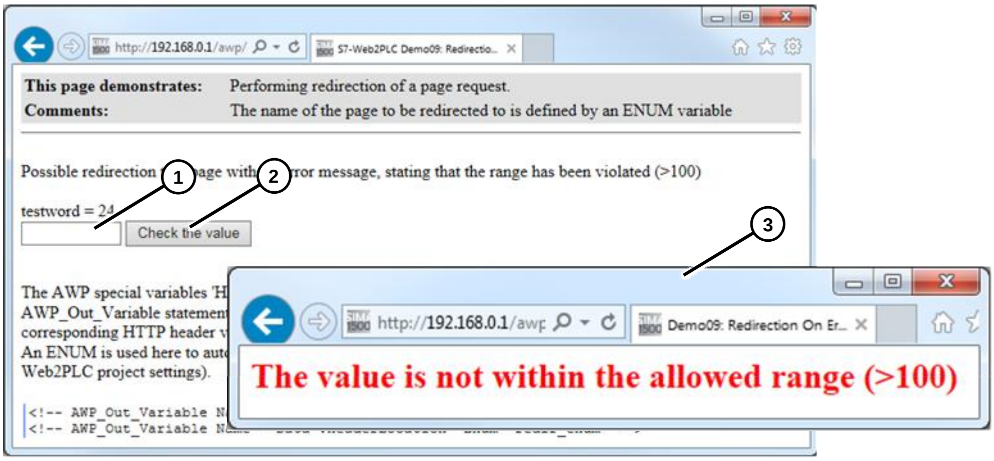
Requirements for the automation task

- Programming a web page on which a tag is written to by means of an input box
- An ENUM tag is to control the (Uniform Resource Locator).
- This ENUM tag is controlled by the STEP 7 program.

9.3 Functional mechanisms

Setup of the web page

Figure 9-1 HTTP redirection web page



This page demonstrates: Performing redirection of a page request.
Comments: The name of the page to be redirected to is defined by an ENUM variable

Possible redirection [illegible] page with [illegible]rror message, stating that the range has been violated (>100)

testword = 24

The AWP special variables 'H
AWP_Out_Variable statement
corresponding HTTP header v
An ENUM is used here to auto
Web2PLC project settings).

```
<!-- AWP_Out_Variable N
<!-- AWP_Out_Variable N
```

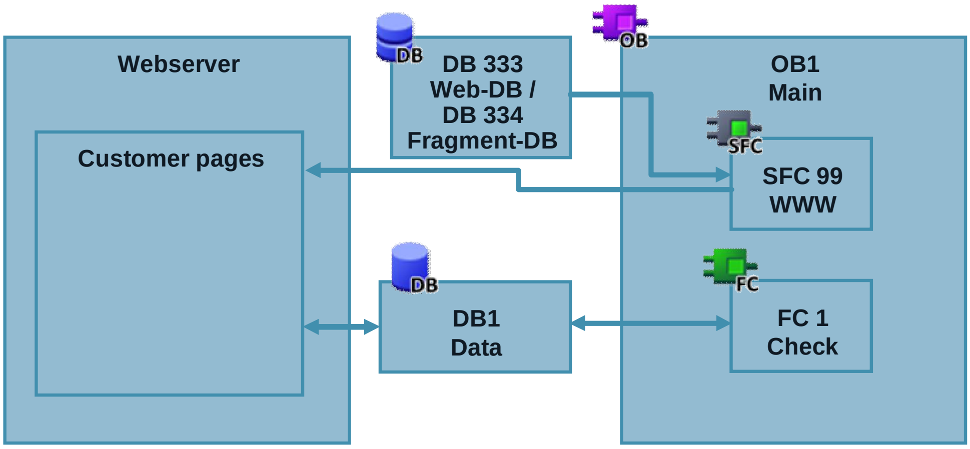
The value is not within the allowed range (>100)

Table 9-1

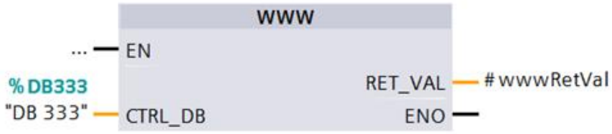
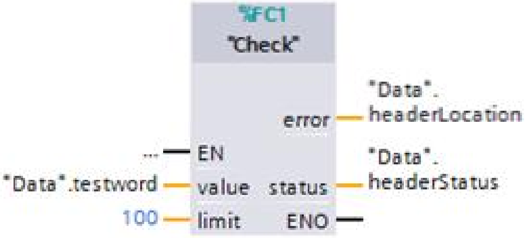
Position number	Description
1	A new value can be entered into the highlighted box.
2	This button hands the value to the controller.
3	If the value transferred to the controller is greater than 100, the selected web page is displayed.

9.3.1 Functional principle of the S7 program

Figure 9-2 S7-program HTTP redirection



Functioning of OB1

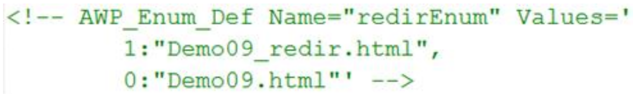
No.	Function
1	 WWW ... EN %DB333 "DB 333" CTRL_DB RET_VAL #wwwRetVal ENO The WWW function (SFC 99) is called here. The web DB (DB 333) is connected to this function. The setup of the user pages is stored in the web DB and in the fragment DB(s). Based on this information, the WWW function (SFC 99) creates the user pages.
2	 %FC1 "Check" error "Data".headerLocation ... EN "Data".testword value status "Data".headerStatus 100 limit ENO The “check” function determines whether the limit has been exceeded. If that is the case, the “headerLocation” tag is set to “1” and “headerStatus” to “302”. This enables switching to the web page with the error message. If the value was not exceeded, both tags are set to “0”.

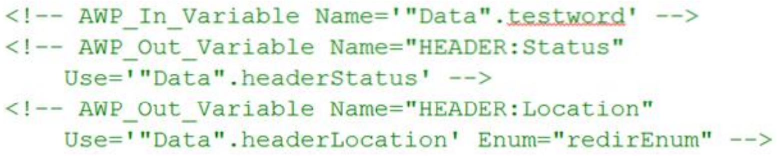
Content of the data DB (DB 1)

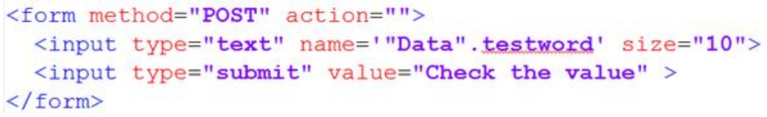
DB 1 stores all tags required for the function.

9.3.2 Functional principle of the HTML file

No.	Function

1	
	`<!-- AWP_Enum_Def Name="redirEnum" Values='`
	`        1:"Demo09_redir.html",`
	`        0:"Demo09.html"' -->`
	To change the location, the corresponding URL has to be assigned to the numbers stored in the "headerLocation" tag. An ENUM tag has to be defined for this purpose.

2	
	`<!-- AWP_In_Variable Name='"Data".testword' -->`
	`<!-- AWP_Out_Variable Name="HEADER:Status"`
	`    Use='"Data".headerStatus' -->`
	`<!-- AWP_Out_Variable Name="HEADER:Location"`
	`    Use='"Data".headerLocation' Enum="redirEnum" -->`
	Initializing the CPU tags and assigning the ENUM tag

3	
	`<form method="POST" action="">`
	`  <input type="text" name='"Data".testword' size="10">`
	`  <input type="submit" value="Check the value" >`
	`</form>`
	Description of the tag to be checked

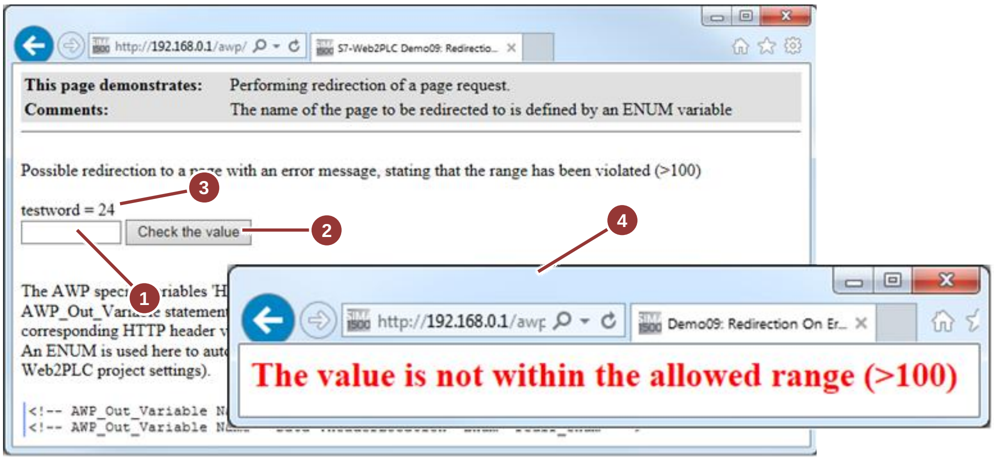
9.4 Operation

Note

> The STEP7 program has the number of this chapter.
>
> To get to the user pages, the “admin” user has to be logged in using the password “s7”. See 3 Principles of Standard Web Pages

1. Enter a new value into the input box.
2. Confirm the button to transfer the value to the CPU.
3. The new value is displayed here.
4. If the value is greater than 100, this web page is displayed subsequently.

Figure 9-3 HTTP redirection web page



This page demonstrates: Performing redirection of a page request.
Comments: The name of the page to be redirected to is defined by an ENUM variable

Possible redirection to a page with an error message, stating that the range has been violated (>100)

testword = 24

The AWP speci riables 'H
AWP_Out_Vari e statement
corresponding HTTP header v
An ENUM is used here to aut
Web2PLC project settings).

```
<!-- AWP_Out_Variable N
<!-- AWP_Out_Variable N
```

The value is not within the allowed range (>100)

Note

> If the checked tag is influenced by the controller, the web page has to be updated cyclically. Otherwise, the HTTP redirect will not be carried out.
>
> To get back to the first page, you can set the value of the PLC tag below one hundred using the TIA Portal and subsequently reload the page in the browser.

10 Language Switching

10.1 Automation task

Language switching (German/English) is to be implemented on a web page.

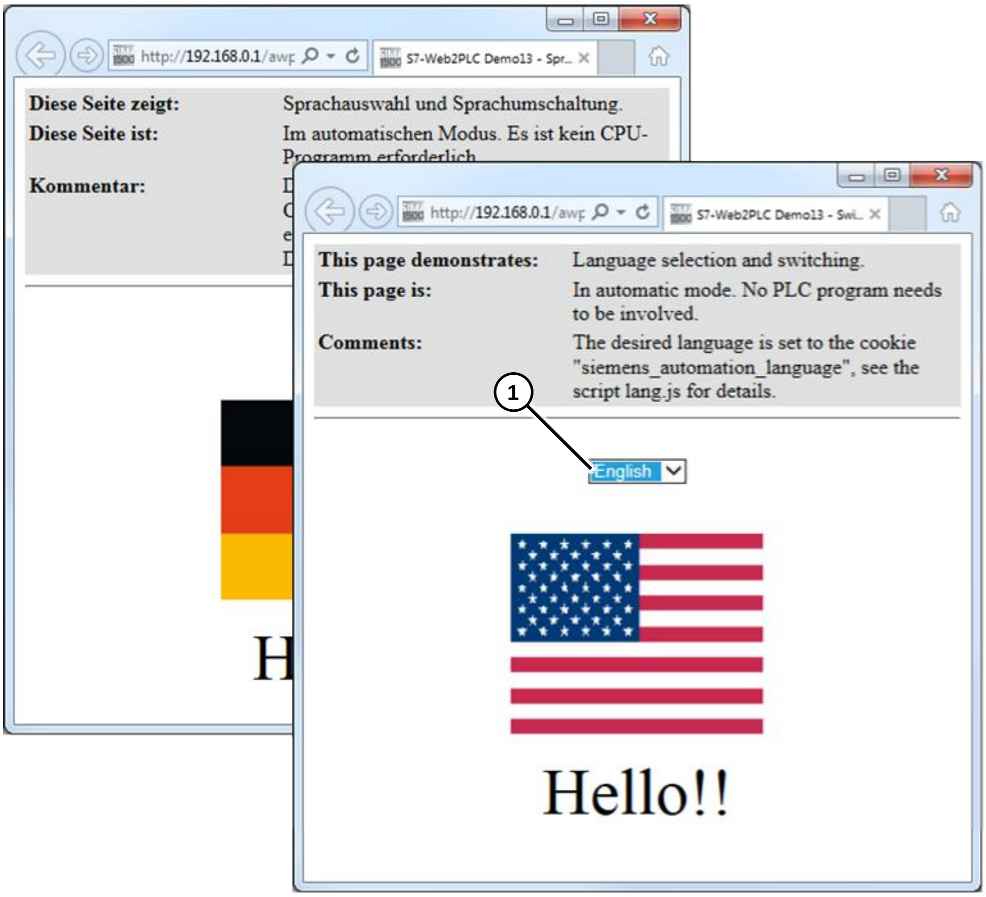
10.2 Automation solution

- Programming two web pages in two different languages
- The language switching is implemented by means of a combo box.

10.3 Functional mechanisms

Setup of the web page

Figure 10-1 language switching web page



Diese Seite zeigt: Sprachauswahl und Sprachumschaltung.

Diese Seite ist: Im automatischen Modus. Es ist kein CPU-Programm erforderlich.

Kommentar: D C e D

This page demonstrates: Language selection and switching.

This page is: In automatic mode. No PLC program needs to be involved.

Comments: The desired language is set to the cookie "siemens_automation_language", see the script lang.js for details.

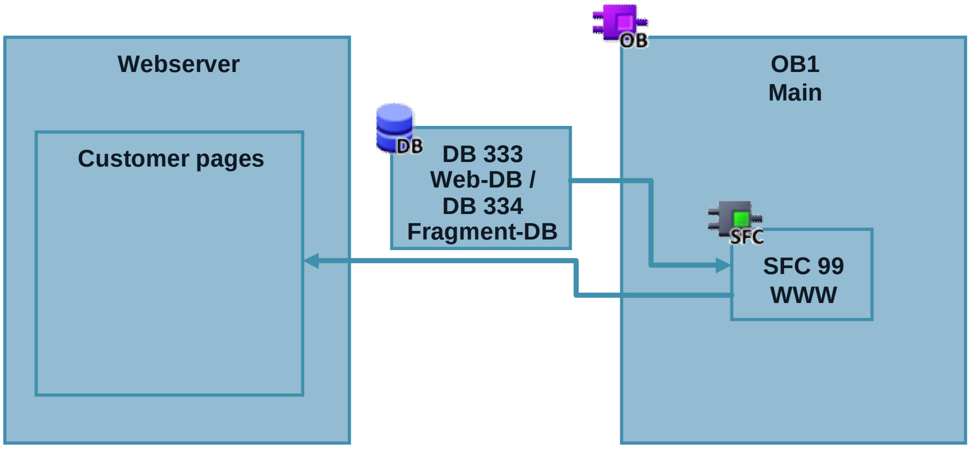
Hello!!

Table 10-1

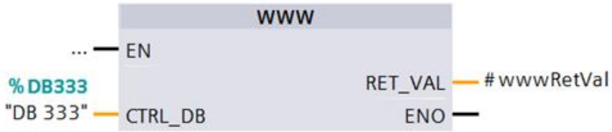
Position number	Description
1	The language is selected in this combo box.

10.3.1 Functional principle of the S7 program

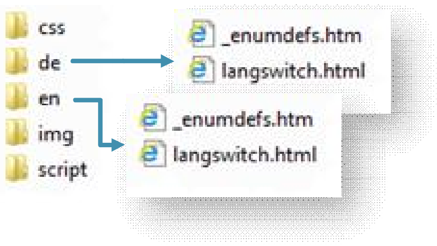
Figure 10-2 S7 program language switching



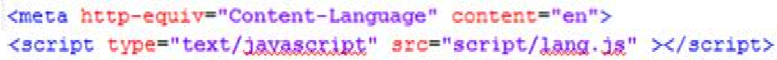
Functioning of OB1

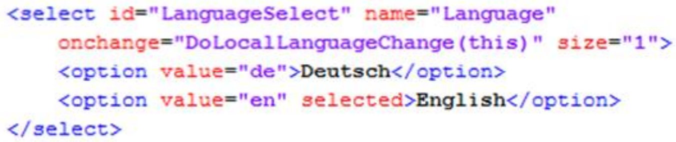
No.	Function
1	 WWW ... EN %DB333 "DB 333" CTRL_DB RET_VAL #wwwRetVal ENO The WWW function (SFC 99) is called here. The web DB (DB 333) is connected to this function. The setup of the user pages is stored in the web DB and in the fragment DB(s). Based on this information, the WWW function (SFC 99) creates the user pages.

10.3.2 Functional principle of the HTML file

No.	Function
1	 css de en img script _enumdefs.htm langswitch.html _enumdefs.htm langswitch.html The structure shown is created in the HTML folder. The respective folder is accessed based on which language is set. Use the following language codes: de : German fr : French en : English it : Italian es : Spanish zh : Chinese The internal structure of the language folders must be the same as that of the other language folders. Otherwise, it won't be possible to access the web pages anymore.

No.	Function
2	The file “lang.js” is located in the “scipt” folder. This JavaScript file contains the function “DoLocalLanguageChange” which controls the language switching.

3	
	`<meta http-equiv="Content-Language" content="en">`
	`<script type="text/javascript" src="script/lang.js" ></script>`
	These two lines can be found in the header of each visible HTML file. “Content-Language” specifies which language is used.
	The JavaScript file “lang.js” controls the language switching.

	
	`<select id="LanguageSelect" name="Language"`
	`    onchange="DoLocalLanguageChange(this)" size="1">`
	`    <option value="de">Deutsch</option>`
	`    <option value="en" selected>English</option>`
	`</select>`
	This is the selection option on the web page.

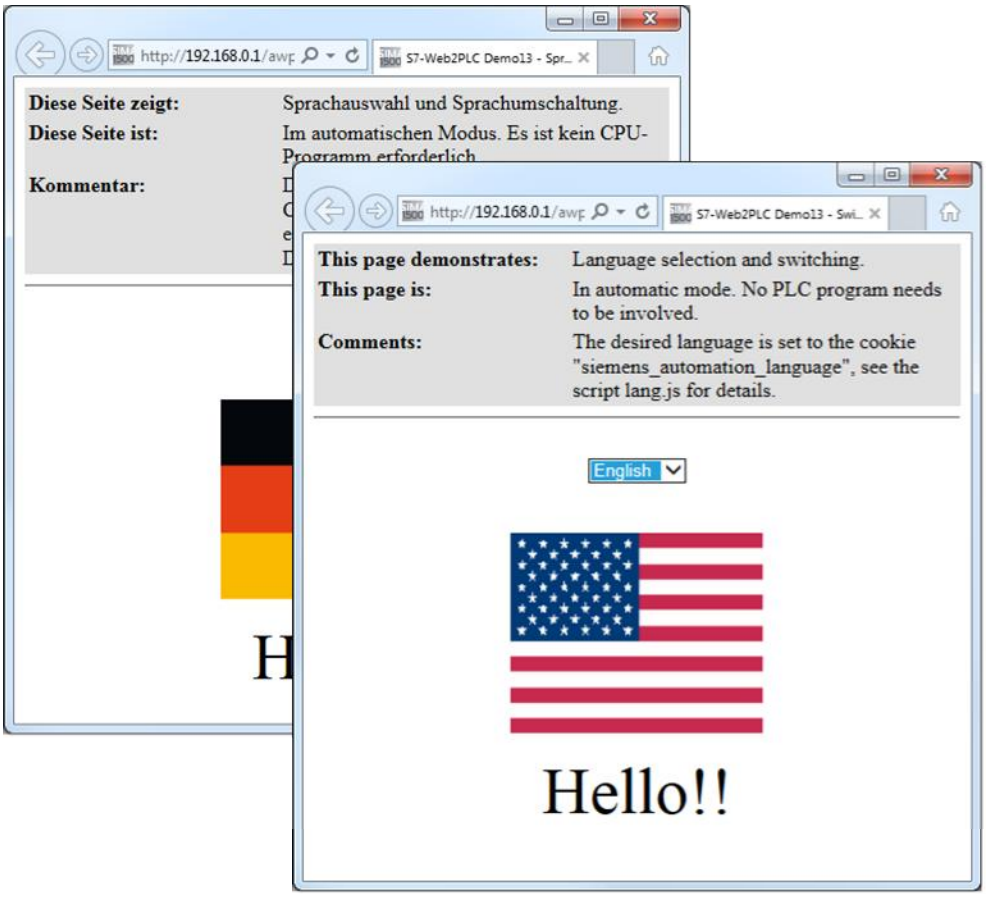
10.4 Operation

Note

> The STEP7 program has the number of this chapter.
>
> To get to the user pages, the “admin” user has to be logged in using the password “s7”. See 3 Principles of Standard Web Pages

Click the combo box to select the desired language.

Figure 10-3 Language switching web page



Diese Seite zeigt:	Sprachauswahl und Sprachumschaltung.
Diese Seite ist:	Im automatischen Modus. Es ist kein CPU-Programm erforderlich
Kommentar:	

This page demonstrates:	Language selection and switching.
This page is:	In automatic mode. No PLC program needs to be involved.
Comments:	The desired language is set to the cookie "siemens_automation_language", see the script lang.js for details.

11 Transmitting Data without Reloading Pages with AJAX

11.1 Automation task

The task involves programming two web pages linked with each other. Both web pages are structured identically. The “velocity” value is read. This value determines the flow. A bar is located underneath. Depending on the “velocity” value, the bar will fill up slowly or quickly.

11.2 Automation solution

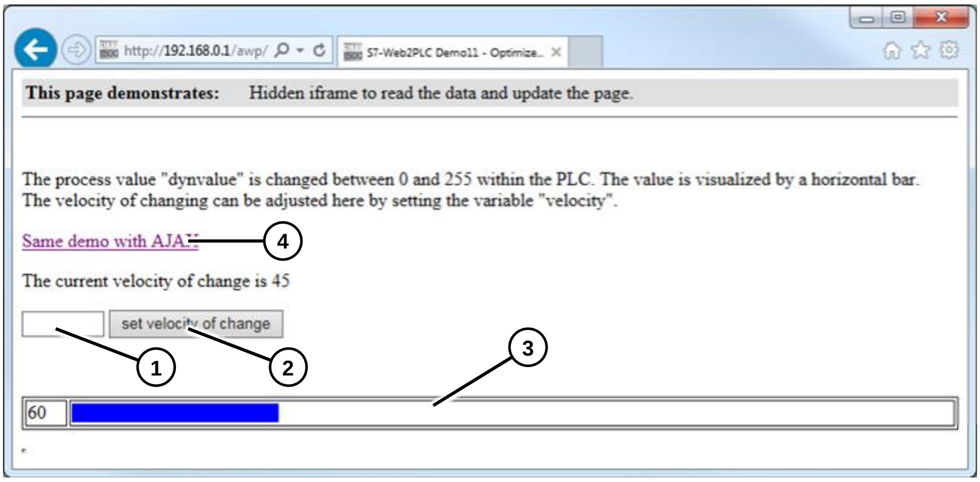
Requirements for the automation task

- Cyclic loading of a value using an inline frame
- Cyclic loading of a value using AJAX
- Writing a value using AJAX
- Writing a value using a form

11.3 Functional mechanisms

Setup of the web page

Figure 11-1 AJAX web page



This page demonstrates: Hidden iframe to read the data and update the page.

The process value "dynvalue" is changed between 0 and 255 within the PLC. The value is visualized by a horizontal bar. The velocity of changing can be adjusted here by setting the variable "velocity".

Same demo with AJAX

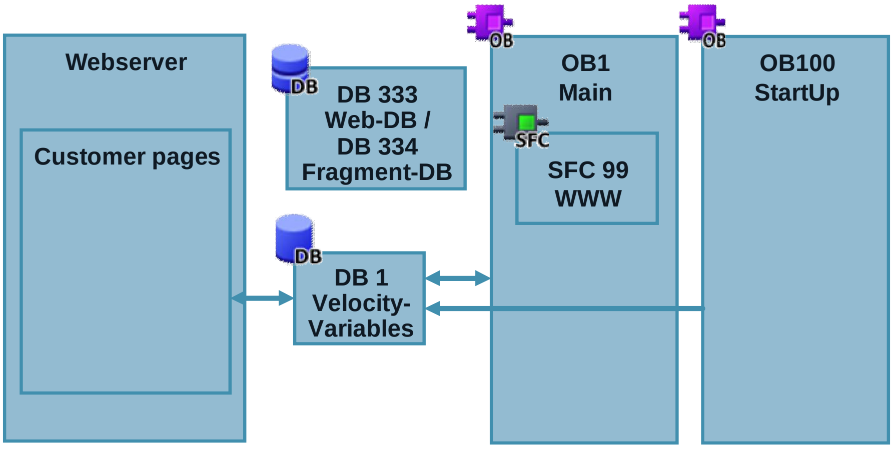
The current velocity of change is 45

Table 11-1

Position number	Description
1	A new flow value can be entered in this field.
2	Click this button to transfer the entered valued to the controller.
3	Depending on the entered value, the bar will fill up slowly or quickly.
4	This link takes you to the AJAX demo.

11.3.1 Functional principle of the S7 program

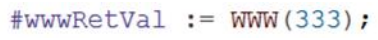
Figure 11-2 S7-AJAX program

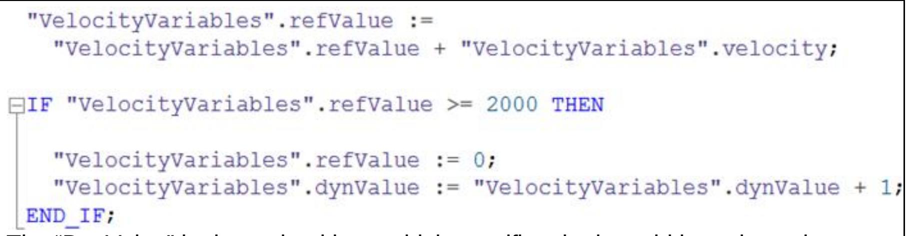


Function of OB100

In OB100, the tags“DynValue” and “Velocity” are reset. “DynValue” determines the bar length. “Velocity” determines the bar filling speed.

Function of OB1

No.	Function
1	 `#wwwRetVal := WWW(333);` The WWW function (SFC 99) is called here. The web DB (DB 333) is connected to this function. The setup of the user pages is stored in the web DB and in the fragment DB(s). Based on this information, the WWW function (SFC 99) creates the user pages.

2	
	`"VelocityVariables".refValue :=`
	`  "VelocityVariables".refValue + "VelocityVariables".velocity;`
	
	`IF "VelocityVariables".refValue >= 2000 THEN`
	
	`  "VelocityVariables".refValue := 0;`
	`  "VelocityVariables".dynValue := "VelocityVariables".dynValue + 1;`
	`END_IF;`
	The “DynValue” is determined here which specifies the bar width on the web page. When “DynValue” reaches 255, the tag has reached the highest value a byte tag can achieve. If the value is increased by one, the byte tag becomes zero.

Content of the velocity tag DB (DB 1)

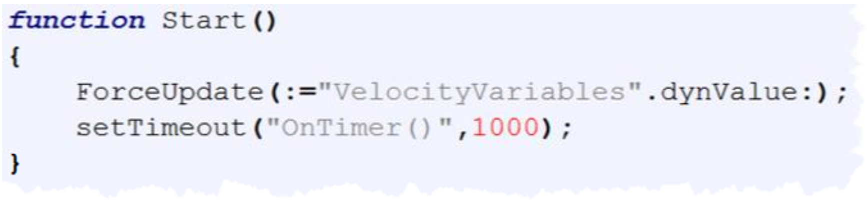
All required tags are stored in this DB.

11.3.2 Functional principle of the HTML file

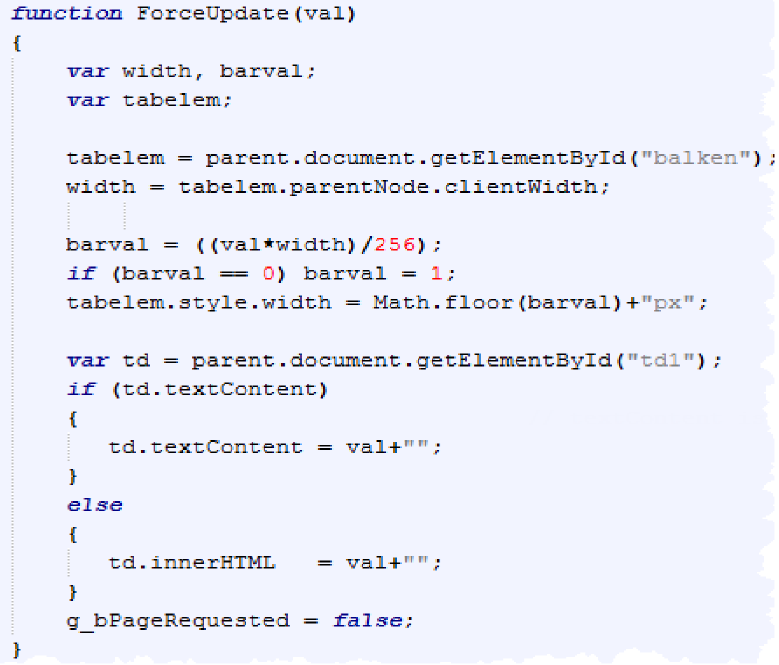
This application example requires HTML knowledge. Mainly the JavaScript code will be explained in the following.

Web page function without AJAX

No.	Function
1	 `<body onload="Start()">` Each time the page has been refreshed, the “Start” function is executed.

2	
	`function Start()`
	`{`
	`    ForceUpdate(:="VelocityVariables".dynValue:);`
	`    setTimeout("OnTimer()",1000);`
	`}`
	Two functions are called in the “Start” function. The “ForceUpdate” function and the “OnTimer” function which is called with a delay of 1000 milliseconds.

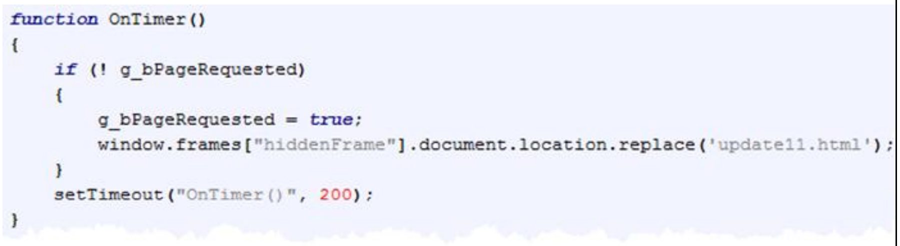
<table>
<tr><th>No.</th><th>Function</th></tr>
<tr><td>3</td><td>



```
function ForceUpdate(val)
{
    var width, barval;
    var tabelem;

    tabelem = parent.document.getElementById("balken");
    width = tabelem.parentNode.clientWidth;

    barval = ((val*width)/256);
    if (barval == 0) barval = 1;
    tabelem.style.width = Math.floor(barval)+"px";

    var td = parent.document.getElementById("td1");
    if (td.textContent)
    {
        td.textContent = val+"";
    }
    else
    {
        td.innerHTML    = val+"";
    }
    g_bPageRequested = false;
}
```

The “ForceUpdate” function determines and transmits the bar length.

</td></tr>
<tr><td>4</td><td>



```
function OnTimer()
{
    if (! g_bPageRequested)
    {
        g_bPageRequested = true;
        window.frames["hiddenFrame"].document.location.replace('update11.html');
    }
    setTimeout("OnTimer()", 200);
}
```

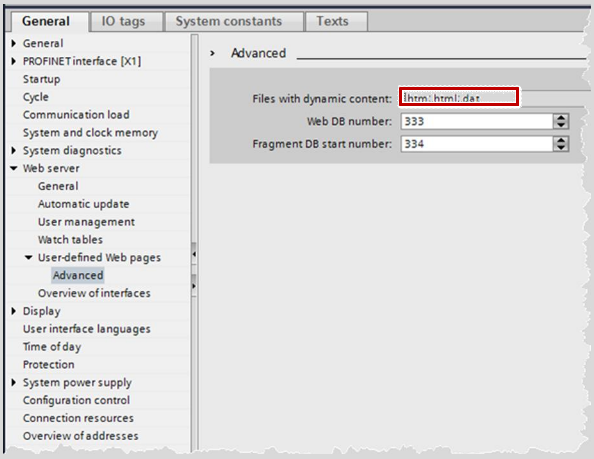
Every 200 ms, the “OnTimer” function updates the value of “DynValue”. This is done using an Iframe which calls the “ForceUpdate” function.

</td></tr>
<tr><td></td><td>To write the “velocity” value, a form is used that is send via the “submit” button.</td></tr>
</table>

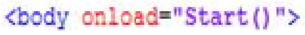
Web page function with AJAX

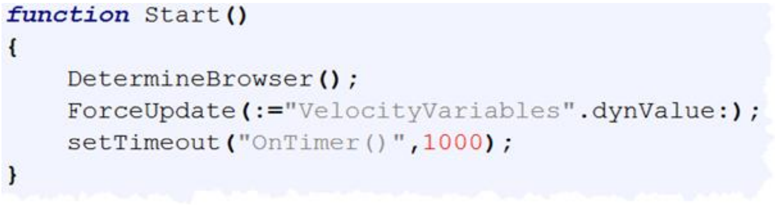
Updating and writing of PLC tags with AJAX is implemented on this web page.

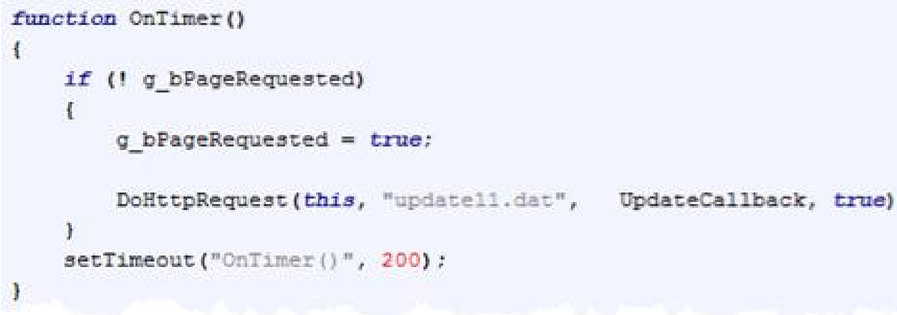
Note For updating with AJAX to work, the “.dat” file format has to be specified as a file format with dynamic content in the CPU settings.



Files with dynamic content: .htm;.html;.dat
Web DB number: 333
Fragment DB start number: 334

No.	Function
1	 `<body onload="Start()">` Each time the page has been refreshed, the “Start” function is executed.

2	
	`function Start()`
	`{`
	`    DetermineBrowser();`
	`    ForceUpdate(:="VelocityVariables".dynValue:);`
	`    setTimeout("OnTimer()",1000);`
	`}`
	Three functions are called in the “Start” function. The “DetermineBrowser” function determines the browser (Mozilla, Internet Explorer, ...). This function can be found in the file “ajaxbase.js”. The “ForceUpdate” function is the same function as described in the section “Web page function without AJAX”. The “OnTimer” function is called with a delay of 1000 milliseconds and updates the values here as well.

<table>
<tr><th>No.</th><th>Function</th></tr>
<tr><td><b>3</b></td><td>


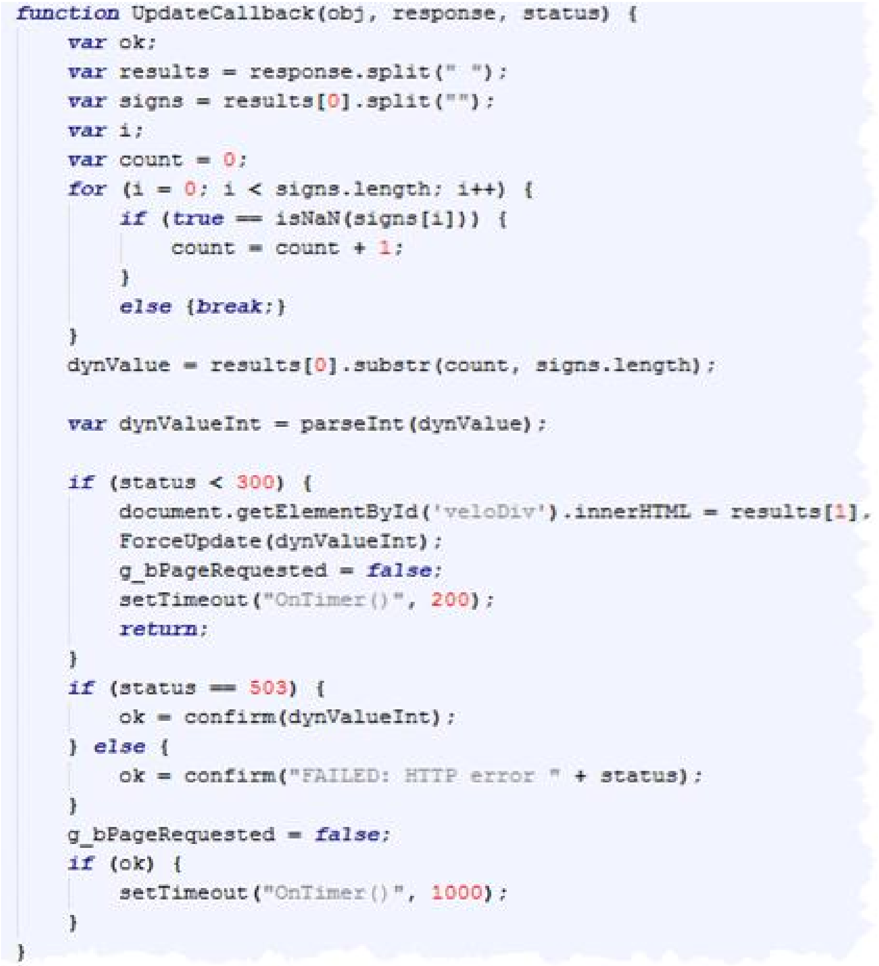
```
function OnTimer()
{
    if (! g_bPageRequested)
    {
        g_bPageRequested = true;

        DoHttpRequest(this, "update11.dat",   UpdateCallback, true)
    }
    setTimeout("OnTimer()", 200);
}
```

Every 200 ms, the “OnTimer” function updates the value. This is accomplished by calling the “DoHttpRequest” function. This function can be found in the file “ajaxbase.js”. Four values are handed over to this function.

6. The object which called the current function.
7. The URL under which to find the tags to be updated. In this case: “update03.dat”.
8. The function which further processes the value and the status code (“UpdateCallback”).
9. Whether data transmission is asynchronous (“true”).

After the values have been updated, the “UpdateCallback” function is called.
</td></tr>
</table>

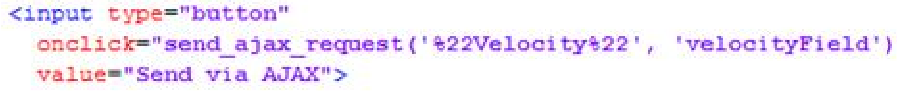
<table>
<tr><th>No.</th><th>Function</th></tr>
<tr><td>4</td><td>



```
function UpdateCallback(obj, response, status) {
    var ok;
    var results = response.split(" ");
    var signs = results[0].split("");
    var i;
    var count = 0;
    for (i = 0; i < signs.length; i++) {
        if (true == isNaN(signs[i])) {
            count = count + 1;
        }
        else {break;}
    }
    dynValue = results[0].substr(count, signs.length);

    var dynValueInt = parseInt(dynValue);

    if (status < 300) {
        document.getElementById('veloDiv').innerHTML = results[1],
        ForceUpdate(dynValueInt);
        g_bPageRequested = false;
        setTimeout("OnTimer()", 200);
        return;
    }
    if (status == 503) {
        ok = confirm(dynValueInt);
    } else {
        ok = confirm("FAILED: HTTP error " + status);
    }
    g_bPageRequested = false;
    if (ok) {
        setTimeout("OnTimer()", 1000);
    }
}
```

The “UpdateCallback” function processes the read values and the status code. The “response” tag contains the updated values of the PLC tags. In order to further process the values, they have to be split and assigned. The “status” variable contains the current HTTP status code. If the HTTP status code is smaller than 300, the “ForceUpdate” function is called. It calculates the bar length from the updated value.

</td></tr>
<tr><td>5</td><td>

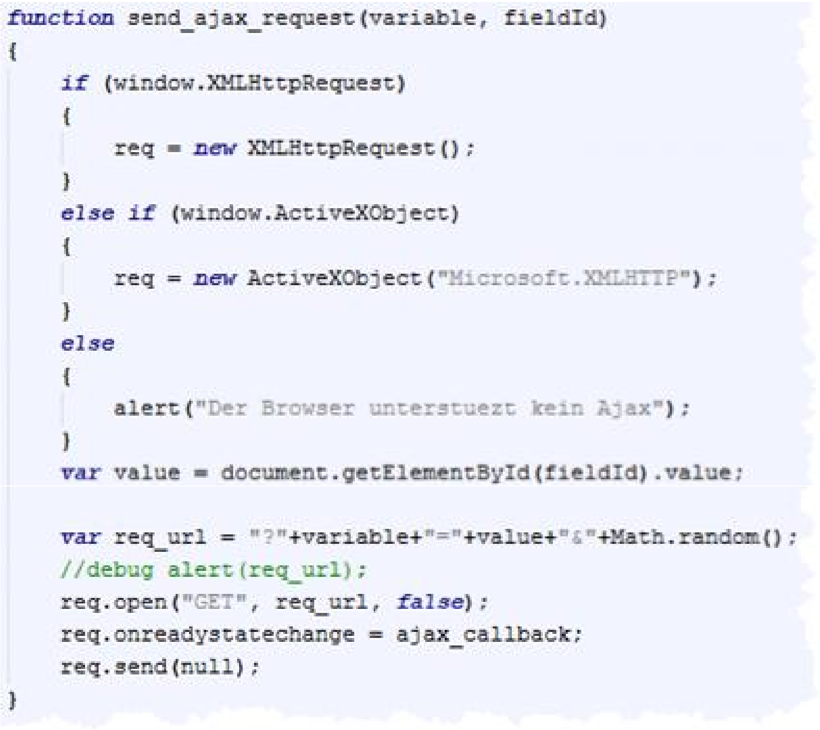


```
<input type="button"
  onclick="send_ajax_request('%22Velocity%22', 'velocityField')
  value="Send via AJAX">
```

By clicking the “Send via AJAX” button, the “send_ajax_request” function is called.

</td></tr>
</table>

<table>
<tr><th>No.</th><th>Function</th></tr>
<tr><td><b>6</b></td><td>



```
function send_ajax_request(variable, fieldId)
{
    if (window.XMLHttpRequest)
    {
        req = new XMLHttpRequest();
    }
    else if (window.ActiveXObject)
    {
        req = new ActiveXObject("Microsoft.XMLHTTP");
    }
    else
    {
        alert("Der Browser unterstuezt kein Ajax");
    }
    var value = document.getElementById(fieldId).value;

    var req_url = "?"+variable+"="+value+"&"+Math.random();
    //debug alert(req_url);
    req.open("GET", req_url, false);
    req.onreadystatechange = ajax_callback;
    req.send(null);
}
```

In the “send_ajax_request” function, the entered value is transferred to the CPU.
</td></tr>
</table>

11.4 Operation

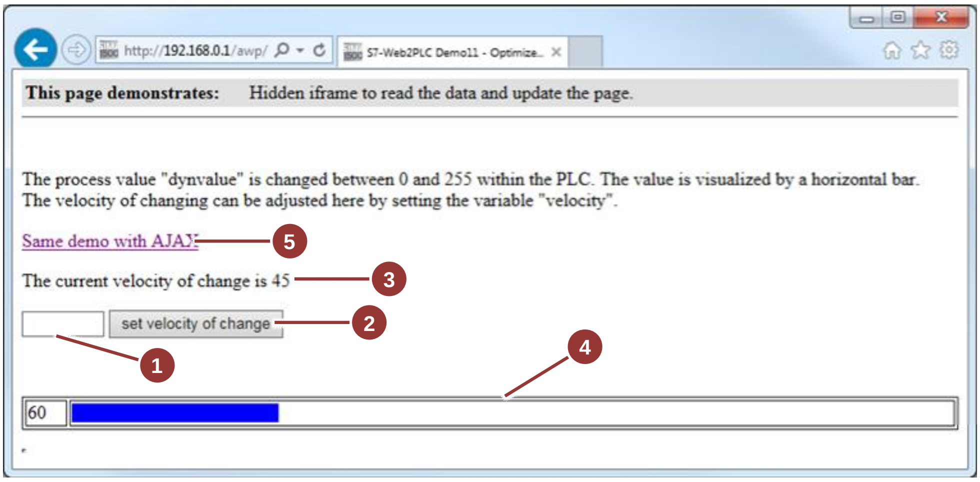
Note

The STEP7 program has the number of this chapter.

To get to the user pages, the “admin” user has to be logged in using the password “s7”. See 3 Principles of Standard Web Pages

1. Enter a new value into the input box.
2. Confirm the button to transfer the new value to the CPU.
3. The new value is displayed here.
4. The bar will now fill more slowly or quickly.
5. This link takes you to the page with the AJAX transfer method.

Figure 11-3 AJAX web page



This page demonstrates: Hidden iframe to read the data and update the page.

The process value "dynvalue" is changed between 0 and 255 within the PLC. The value is visualized by a horizontal bar. The velocity of changing can be adjusted here by setting the variable "velocity".

Same demo with AJAX

The current velocity of change is 45

12 Recording a PLC Tag with a Graph

12.1 Automation task

The task is to program a web page showing a graph. This graph is to illustrate a PLC tag graphically.

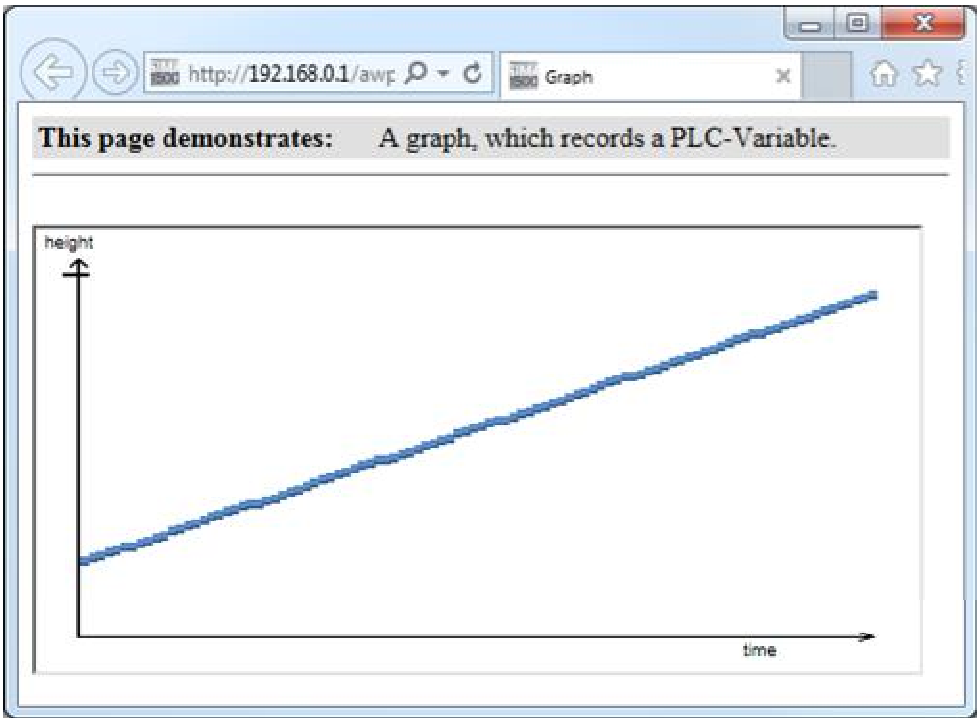
Requirements for the automation task

- Cyclic reading of a PLC tag using an Iframe.
- Saving the previous values of the PLC tag in cookies.
- Processing the values in a JavaScript file.
- Incrementing and decrementing a tag in the STEP 7 program

12.2 Functional mechanisms

Setup of the web page

Figure 12-1

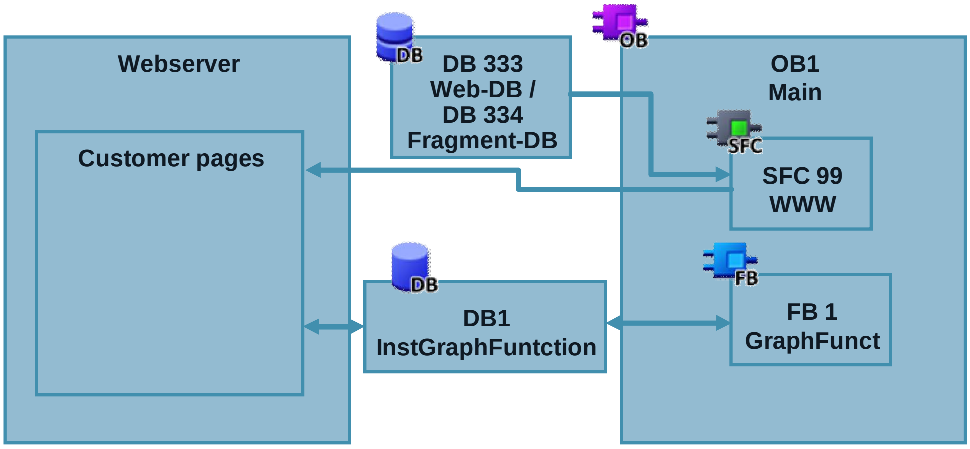


This page demonstrates: A graph, which records a PLC-Variable.

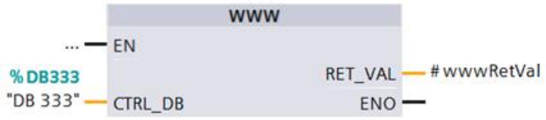
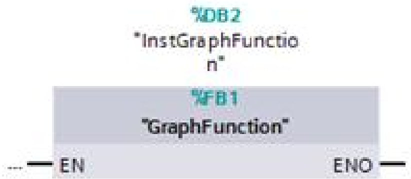
A graph showing the graphical development of a PLC tag is displayed.

12.2.1 Functional principle of the S7 program

Figure 12-2 S7- graph program



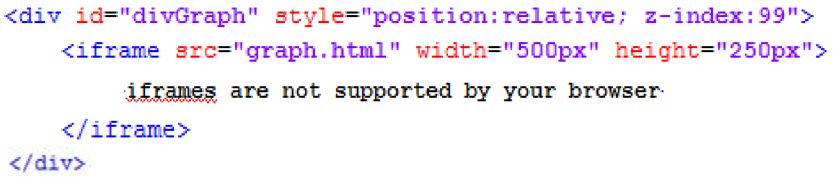
Function of OB1

No.	Function
1	 WWW ... EN %DB333 "DB 333" CTRL_DB RET_VAL #wwwRetVal ENO Calling the WWW function (SFC99) initializes the web server of the CPU. By calling the function cyclically, the web page always shows the latest values of the PLC tags.
2	 %DB2 "InstGraphFunctio n" %FB1 "GraphFunction" ... EN ENO The tag “statGraphVariable” is incremented and decremented in FB1 to reflect changes on the graph.

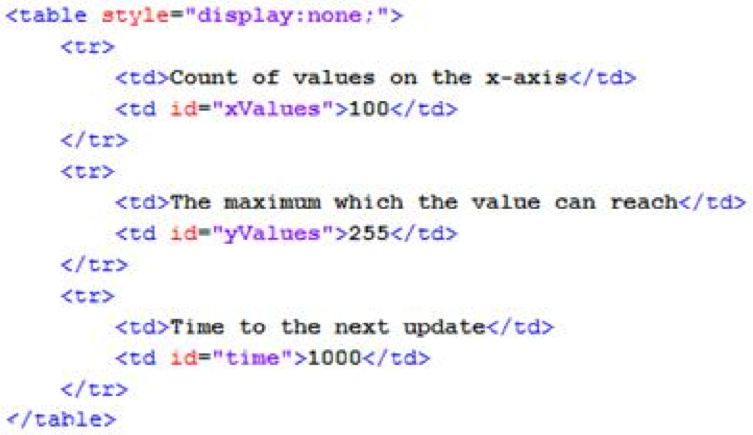
12.2.2 Functional principle of the HTML file

Function of the web page

No.	Function

1	
	`<div id="divGraph" style="position:relative; z-index:99">`
	`<iframe src="graph.html" width="500px" height="250px">`
	`iframes are not supported by your browser`
	`</iframe>`
	`</div>`
	The “Demo12.html” HTML file has an Iframe integrated. The graph size can be adjusted via the properties “width” and “height”. You will see a message if your browser does not support Iframes.

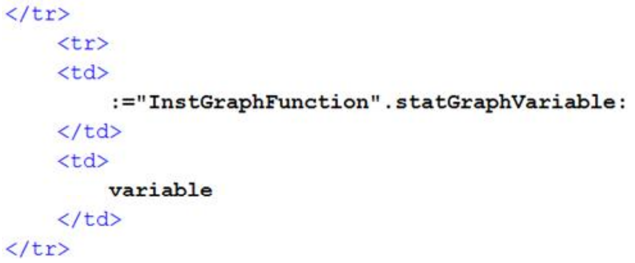
<table>
<tr><th>No.</th><th>Function</th></tr>
<tr><td><b>2</b></td><td>



```
<table style="display:none;">
    <tr>
        <td>Count of values on the x-axis</td>
        <td id="xValues">100</td>
    </tr>
    <tr>
        <td>The maximum which the value can reach</td>
        <td id="yValues">255</td>
    </tr>
    <tr>
        <td>Time to the next update</td>
        <td id="time">1000</td>
    </tr>
</table>
```

In the “graph.html” HTML file, you can make various settings in a table. In the first row you can specify how many values are to be recorded in parallel. In the second row you can specify how large the highest value of the PLC tag to be recorded can become. In the third row you can specify after which time the values are updated.

The JavaScript file “graph.js” accesses the values in the table.</td></tr>
<tr><td><b>3</b></td><td>



```
</tr>
    <tr>
    <td>
        :="InstGraphFunction".statGraphVariable:
    </td>
    <td>
        variable
    </td>
</tr>
```

In the “Update_graph_value.html” HTML file you can enter which PLC tag to record in a table.</td></tr>
<tr><td><b>4</b></td><td>The Java scrip file “graph.js” is integrated in the “graph.html” file.

The read in new values are stored in cookies in the JavaScript file. Subsequently, all values are read from the cookies and the percentage position of the points is calculated in the Iframe.</td></tr>
</table>

12.3 Operation

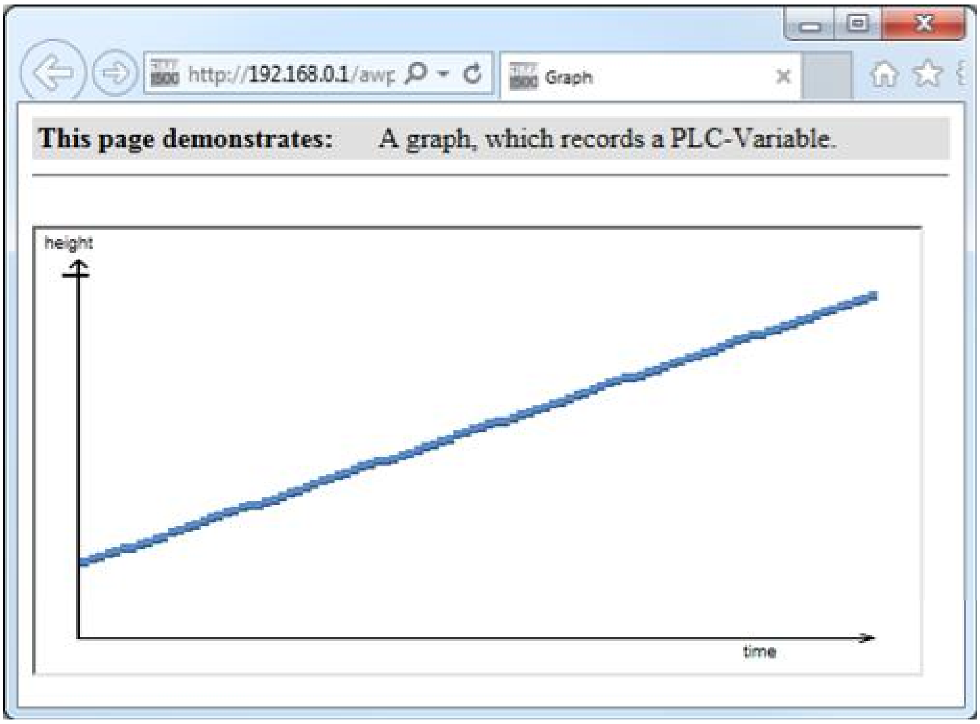
Note

The STEP7 program has the number of this chapter.

To get to the user pages, the “admin” user has to be logged in using the password “s7”. See 3 Principles of Standard Web Pages

The graph is displayed on the web page.

Figure 12-3



This page demonstrates: A graph, which records a PLC-Variable.

13 Configuration and Settings

In the application example “Creating and using user-defined web pages on S7-1200 / S7-1500” you will find all the information on how to create and operate a web page for a CPU with PN interface yourself.

14 Installation

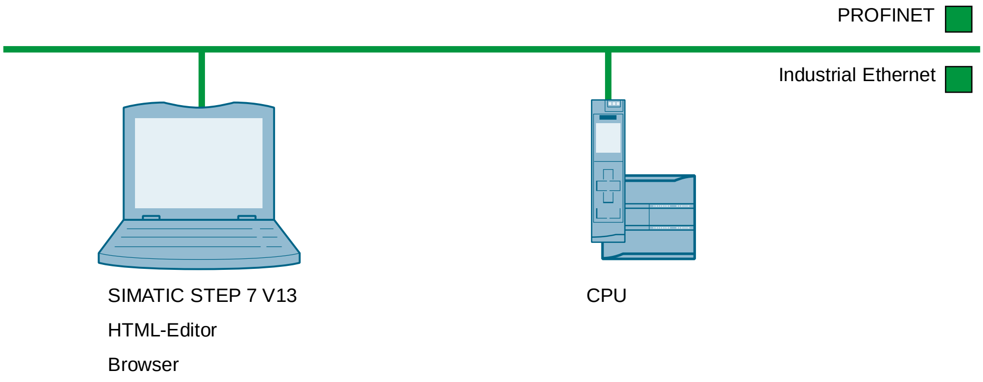
14.1 Hardware and software installation

Hardware installation

The figure below shows the hardware configuration of the examples.

The PC including web browser has to be connected to the CPU via Industrial Ethernet through the PN interface.

Figure 14-1



Note Please observe the installation and connection guidelines from the corresponding manuals.

Installing the software

Table 14-1

No.	Action	Remarks
1.	Install SIMATIC STEP 7 Professional (TIA Portal).	
2.	Install a tool for creating the web page, e.g. MS Frontpage or Notepad++ on the PC on which you want to create the web page.	
3.	Install a web browser on the PC, e.g. Firefox or Internet Explorer, which you want to use to access the web page of the CPU.	

14.2 Installing the application example

Table 14-2

No.	Action	Remarks
4.	Unzip the file "68011496_simple_examples_for_webserver_CODE_v10.zip " file in your project directory.	
1.	Start SIMATIC STEP 7 V13	
2.	Open the project in SIMATIC STEP 7 V13.	
3.	Select the desired program. The programs are numbered by chapter numbers in the documentation.	
4.	Go to the device view.	
5.	If you are using a different CPU, change the device.	When replacing S7-1200, web server settings may be lost. These settings have to be re-entered afterwards. Check the following setting: Properties (CPU) -> Web server -> Overview of interfaces -> Enabled web server access.
6.	In the CPU properties of the Ethernet interface, assign the IP address of your CPU.	
7.	Select the CPU and load the entire project in the CPU.	
8.	Start a web browser and call the web page of your CPU via the IP address.	For more information, see the examples in the chapter "Operation".

15 Related Literature

15.1 Bibliographic references

This table offers you a variety of pertinent literature.

Table 15-1

No.	Topic	Title
/1/	HTML	HTML und CSS, Praxisrezepte für Einsteiger Robert R. Agular mitp ISBN 978-3-8266-1779-9
/2/	HTML	HTML Handbuch Stefan Münz/Wolfgang Nefzger Franzis Verlag ISBN 3-7723-6654-6
/3/	Javascript	JavaScript und Ajax, Das umfassende Handbuch Christian Wenz Galileo Press ISBN 978-3-8362-1128-4

15.2 Internet link specifications

This table contains a selection of links on further information.

Table 15-2

No.	Topic	Title
/1/	Link to this document	https://support.industry.siemens.com/cs/ww/en/view/68011496
/2/	Siemens Industry Online Support	http://support.automation.siemens.com
/3/	HTML, JavaScript	http://www.selfhtml.de/ http://de.selfhtml.org/ http://www.little-boxes.de/
/4/	S7-1500 web server function manual	https://support.industry.siemens.com/cs/ww/en/view/59193560

16 History

Table 16-1

Version	Date	Modifications
V1.0	08/2015	First version